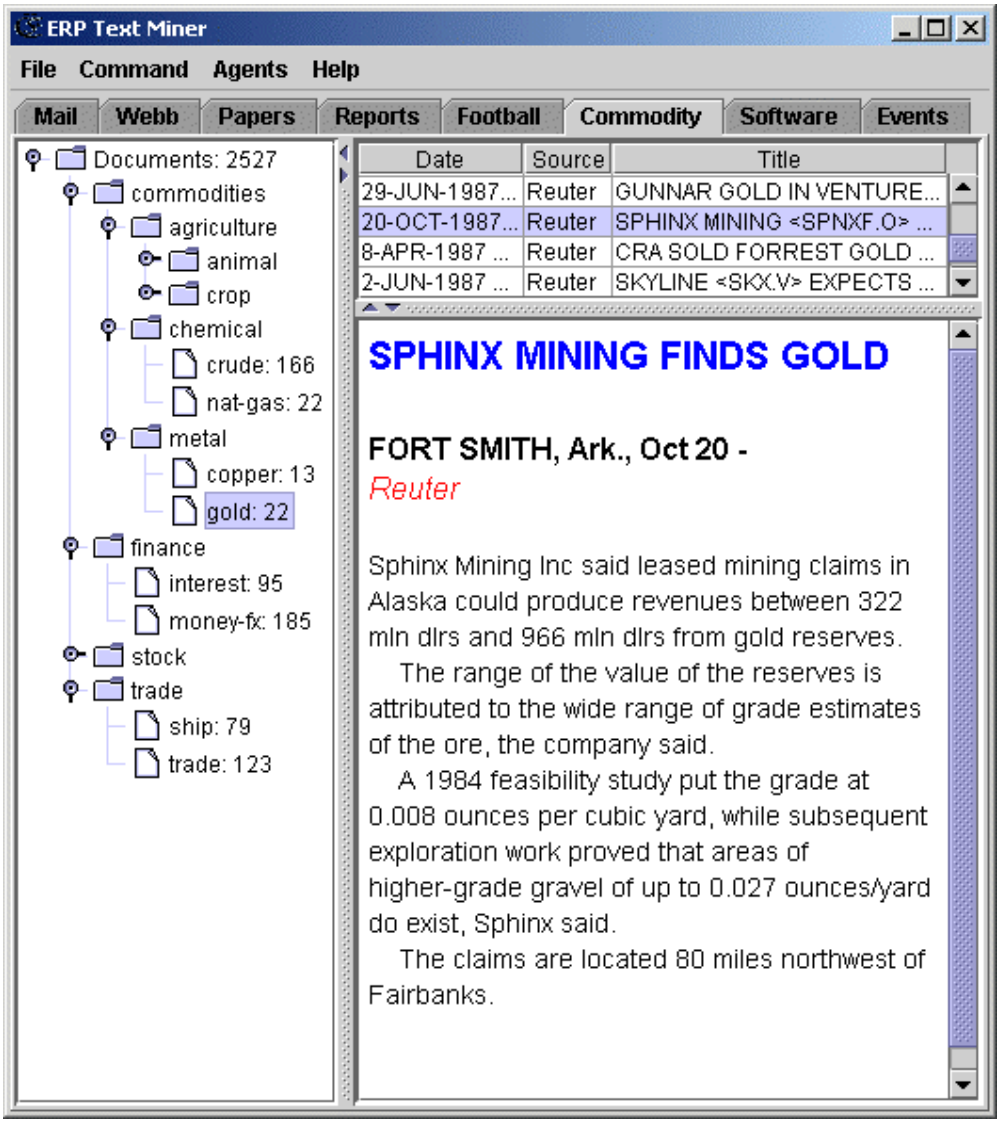
HANDELSHÖGSKOLAN
GÖTEBORGS UNIVERSITET
Institutionen för Informatik
Magisteruppsats den 1 december 2003
Erik Plenk

ERP Text Miner: ett agentsystem för automatisk textklassificering



Sammanfattning
Mängden information lagrad on-line är enorm och fortsätter att växa i en nästan exponentiell hastighet. Webbsidor är bara ett exempel på dokument som behöver hittas, hämtas, bläddras och läsas: användare förväntar sig att andra typer av dokument – som elektronisk mail, dokumentation, och nyheter – också ska vara lagrad och tillgänglig on-line. Men det blir alltmer svårare att finna det man söker. Sökmaskiner är många gånger otillräckliga, antigen får man tiotusen träffar eller ingen allas. Det behövs bättre verktyg för att lösa problemet. Automatisk textklassificering är uppgiften att utveckla datorprogram som kan organisera dokument i kategorier. Med en välordnad semantisk struktur kan man ”borra” sig ned till den information man efterfrågar. Konceptet agent är ett annat viktigt område. Teknologin har börjat uppfattas som helt avgörande, inte endast för att hantera den ökande informationen, utan också som utvecklingsmodell där man kan utnyttja den effektivitet som kännetecknar organiserat beteende. Detta arbete utforskar teknologierna automatisk textklassificering och agenter. De olika stegen i klassificeringsprocessen beskrivs. Resultat presenteras från utförda experiment med maskininlärningsmetoden k-Nearest Neighbor och kollektionen Reuters-21578. Ett agentsystem för automatisk textklassificering utvecklas. Användningen av systemet beskrivs. Arbetet ger en vägledningsmodell för utvecklare som önskar kombinera teknologierna automatiskt textklassificering och agenter.

Nyckelord
Agent, automatisk textklassificering, maskininlärning, systemutveckling, information overload

Författare: Erik Plenk
Handledare: Faramarz Agahi
Magisteruppsats, 20 poäng

INNEHÅLLSFÖRTECKNING

TABELLER

FIGURER

1 INTRODUKTION

1.1 Bakgrund

En enorm volym av dokument flödar genom organisationer varje dag: webbsidor, email, rapporter, artiklar, kundunderöversikter, patentinformation, undersökningar, resuméer och mycket mer. Det finns inte tid att läsa allt, än mindre att skapa någon form av ordning så att värdefull information kan tillvaratas.

Vi måste lära oss att navigera i denna nya ocean av data. Detta kan vara ett verkligt hinder då våra navigeringsstöd som sökmaskiner fortfarande är primitiva och ineffektiva. Väldigt ofta när vi försöker hitta ett stycke text returnerar sökmaskinen hundra, tusen, kanske även hundra tusentals träffar. Vi misstänker att den informationen oumbärlig för oss troligtvis är nedgrävd mellan de returnerade sidorna – men var exakt?

Det skulle vara mycket enklare att hantera denna explosion av information, att smälta all data som svämmar över oss, om vi kunde identifiera huvudämnena i informationen och ytterligare organisera dessa ämnen i någon typ av struktur, helst hierarkisk.

Textklassificering är ett sätt att skapa en sådan struktur. Textklassificering organiserar dokument eller webbsidor i logiska grupper baserat på deras innehåll. Idealet är att de dokument som diskuterar samma ämne grupperas tillsammans i kategorier. En studie[1] visade att användare upplever detta mycket fördelaktigt, då det möjligt att hitta relevant information på kortare tid.

Yahoo[2] är en portal som erbjuder sina användare dokument klassificerade i kategorier. Yahoo har logiskt grupperat miljoner webbsidor i lämpliga kategorier och undergrupper. Till exempel, om en användare vill hitta information om ”IFK Göteborgs fotbollsklubb”, är detta möjligt genom att bläddra från ”roten” på portalen till ett ”löv” som innehåller relevant resurs. I detta exempel skulle användaren följa en hierarki med kategorier genom att klicka på länkarna "Sport och fritid", ”Sport”, ”Fotboll”, ”Klubbar”, ”Allsvenskan” och slutligen ”IFK Göteborg”. Alltså, smidigt, enkelt och användarvänligt.

Men att klassificera dokument för hand förbrukar stora resurser. Vid Yahoo sitter 200 anställda och klassificerar dokument i 500 000 kategorier. MEDLINE spenderar 2 miljoner dollar årligen för att klassificera medicinska artiklar i 18 000 kategorier. USA:s folkräkningsbyrå, använder 232 stycken branschkategorier och 504 stycken yrkeskategorier och spenderar 15 miljoner dollar för sin klassificeringsprocess, vilken helt görs för hand[3].

Manuell klassificering är inte enbart tidskrävande och kostsam. Den förlitar sig på subjektiva bedömningar hos människor, den är inte skalbar, och den är också opraktisk vid stora dokument-kollektioner.

Detta examensarbete handlar om utvecklingen av ERP Text Miner. Genom kombinera teknikerna textinlärning och agenter ville jag utforska möjligheten att utveckla system som automatisk kan klassificerar text i kategorier. Ett sådant system skulle kunna underlätta den manuella klassificeringen och i slutändan hjälpa sina användare att snabbare och enklare hitta relevant information.

[1] Chen & Dumais: “Bringing order to the web: Automatically categorizing search results”
[2] www.yahoo.com
[3] Bennet: “Introduction to Text Categorization”

1.2 Automatisk textklassificering

Automatisk textklassificering (ATC) är sysselsättning att utveckla datorprogram med förmåga att klassificera dokument i fördefinierade kategorier. ATC har under senare tid blivit mycket uppmärksammat, beroende på den enorma mängden information och det medföljande behovet att organisera denna för enklare användning. ATC har visat sig vara ett mycket bra komplement till manuell klassificering. Stora informationsvolymer kan struktureras precist, enkelt, och effektivt.

För att utveckla automatiska textklassificerare är maskininlärning den idag dominerande metoden.
Vid maskininlärning utvecklas ett program som automatisk tar fram en klassificerare. Detta görs genom att programmet ”observerar” karaktäristiken hos en uppsättning dokument som tidigare manuellt har klassificeras av en expert. Utifrån denna förvärvade kunskap kan programmet sedan bedöma om ett nytt dokument skall klassificeras under olika kategorier eller inte.

I detta arbete undersöks denna teknik närmare. Jag utvecklare en automatisk textklassificerare vars prestanda utvärderas med en större dokumentkollektion. Vidare visar jag också hur automatisk klassificeringsteknik kan användas i ett agentbaserat system.

1.3 Agenter

Agenter är datorprogram som utför specialiserade uppgifter för sina användare och som kan agera enskilt eller tillsammans med andra agenter, för att förverkliga uppsatta mål. Agenter har en uppsättning egenskaper som särskiljer dom från andra program. Dom egenskaper som är relaterade till mitt arbete är autonomi, reaktivitet, proaktivitet, kontinuerlighet, och samarbetsförmåga.

Att använda agentmetaforen vid systemutveckling är mycket fördelaktigt. Agenter medger ett naturligt sätt att bygga upp system i moduler, dom gör det enklare att konceptualisera, designa och att implementera system. Agenter kapslar in programmeringstekniska detaljer i sig själva, och användaren behöver endast känna till servicen som dom erbjuder.

Detta arbete utforskar agenttekniken. Jag designar och utvecklar en serie agenter som hjälper sina användare att förvärva, förvalta och klassificera information.

1.4 Syfte

Syftet med detta examensarbete är att:

- Förvärva och presentera kunskap om teknikerna automatisk textklassificering och agenter, för att ge förståelse för system som tillämpar dessa tekniker kan utvecklas, utvärderas, kombineras, och i slutändan användas till att snabbare och enklare hitta relevant information.

Den främsta målgruppen för denna undersökning är systemutvecklare som letar efter lösningar som tacklar problemet med den växande informationsmängden. Arbetet avser att ge en praktisk vägledningsmodell till dessa utvecklare. Med ordet ”praktisk” menas att arbetet avser att behandla de flesta aspekter som en utvecklare behöver ta hänsyn till när dessa teknologier tillämpas och kombineras vid en utvecklingsuppgift. Det finns redan en hel del arbeten som berör enskilda tekniker och utvecklingssteg ganska ingående, men däremot väldigt få arbeten med avsikt att behandla hela utvecklingsförlopp. Detta arbete avser således att svara upp mot detta behov.

1.5 Frågeställningar

Utifrån ovanstående syfte har två frågeställningar framträtt. De frågeställningar som kommer att dominera arbetet är:

- Hur utvecklas och utvärderas automatiska textklassificerare, samt vilken prestanda är möjlig att uppnå med en sådan klassificerare?

- Vad kännetecknar en god design för ett system som förvärvar och klassificerar information, samt hur kan man använda ett system som är utvecklat med en sådan design?

1.6 Disposition

Arbetet är uppdelat i 6 kapitel vilka följer introduktionen. Kapitel 2 redogör för metod att ta reda på nödvändig information samt sättet att arbete med insamlat material. I kapitel 3 beskrivs centrala begrepp, utvecklingssteg och utvärderingsmetoder inom automatisk textklassificering. Kapitel 4 redogör bakomliggande teori om agenter som väglett vid framställningen av ERP Text Miner. I kapitel 5 beskrivs dom experiment som jag har genomfört med metoden k-Nearest Neighbor och kollektionen Reuters-21578. Kapitel 6 redogör för utvecklingen av ERP Text Miner, där systemet också demonstreras med ett användarscenario. Slutligen i kapitel 7 sammanfattas och diskuteras slutsatserna för arbetet.

2 METOD

2.1 Inledning

I detta kapitel redogörs för sättet att ta reda på nödvändig information samt sättet att arbeta med insamlat material för uppsatsens framställning. Begrepp som validitet och reliabilitet kommer också att diskuteras.

2.2 Ansats

Vid en undersökning brukar man ta ställning till dessa ansats, eller kunskapssyfte, vilket väljs beroende på hur mycket kunskap som redan finns eller vad undersökningen syftar till. Här diskuteras de ställningstaganden som jag gjorde för mitt arbete.

Patel & Davidson[4] skiljer mellan följande ansatser: explorativa, deskriptiva och hypotesprövande. När det finns luckor i kunskapen om det aktuella ämnet kommer undersökningen att vara utforskande, det vill säga explorativ. Syftet med en sådan undersökning är att inhämta så mycket ny kunskap som möjligt om ett bestämt problemområde. När det redan finns en viss mängd kunskap inom området, och när man börjar systematisera denna i former av modeller, är undersökningen beskrivande, det vill säga deskriptiv. När kunskapsmängden blivit ännu mer omfattande och teorier utvecklats, är det vanligt att man ställer upp kvalificerade gissningar. Sådana undersökningar är då hypotesprövande.

Det första jag ville undersöka var hur automatiska textklassificerare kan utvecklas, utvärderas samt vilken prestanda som är möjligt att uppnå med denna teknologi. Då det redan finns en hel del kunskap inom detta område, ville jag främst skaffa en bättre överblick genom en beskrivning. Ansatsen var således deskriptiv.

Det andra som jag ville undersöka var vad som kännetecknar en god design för ett system som förvärvar och klassificerare information, samt hur man kan använda ett sådant system. Under detta arbete experimenterade jag med olika designförslag vilka var baserade på agentmetaforen.
Vad beträffar området agenter så finns det redan en hel del kunskap. Denna kunskap ville jag då först skapa en bättre överblick genom en beskrivning. Ansatsen var alltså deskriptiv. Men när det handlar om design och användning av agentbaserade klassificeringssystem så är kunskapsläget mycket knappt, eller snarare obefintligt. Den undersökning som vidtog här var alltså i huvudsak explorativ.

2.3 Val av metod

Informatik är en relativ ny mångdisciplinär vetenskap med en mångfald av olika metoder. På grund av denna anledning finns det ingen given metod hur forskning kan bedrivas med informatik som infallsvinkel. Dahlbom[5] säger att det kanske inte är så intressant exakt hur du genomför din forskning, men att du gör det med informationsteknologins användning i åtanke. ”Informatik är inte som naturvetenskapen med sitt uttryckliga intresse för naturen eller samhällsvetenskapen som inte vågar närma sig teknologin.”

Val av metod gjordes utifrån följande övervägande: Informatik är ett designämne med målet att förbättra användningen av IT genom att bidra till utvecklingen av både användning och teknologin som sådan[6].

[4] Patel & Davidson: ”Forskningsmetodikens grunder”
[5] Dahlbom: “The New Informatics”
[6] Dahlbom & Mathiassen: “Computers in Context: The philosophy and practice of systems design”

Informatik studerar alltså relationen IT och dess användning. Detta säger underförstått, att studier av varje individuell komponent i denna relation (teknologi eller användning), så väl som studier av kombinationen (teknologi i användning) är nödvändiga. En aspekt av en sådan studie är då utvecklingen och utvärderingen av IT-verktyg (prototyper) för att få en förståelse av deras potentiella användning. Det är inom denna kategori av undersökningsmetod som jag vill placera mitt arbete.

Den metoden som jag har tillämpat har inneburit utvecklingen av en serie prototyper där teknikerna automatisk textklassificering och agenter kontinuerligt har utvärderats. Det var genom dessa prototyper som jag fick empiriskt kontakt och som i huvudsak försåg mig med material för de frågeställningar som jag hade formulerat. Prototyperna var grundade i sammanställd kunskap från fältet, dels i vissa antaganden som jag gjorde om uppgiftens natur och ett tänkt system. Under arbetets gång gjordes nya antaganden och bedömningar utifrån kvalitén hos prototypen vilket lade grund för efterföljande versioner.

2.4 Val av material

Material kommer från fyra källor: egna utvecklade javapaket, paketet ir, testkollektionen Reuters-21578, samt javapaketet CIAgent. Använt skriftligt material återges i slutet av uppsatsen.

Typiska steg i en textklassificeringsprocess är: förbehandling, representation, termviktning, dimensionsreducering, maskininlärning, samt utvärdering. För att kunna undersöka dessa steg sammanställde jag ett javapaket. Centrala klasser i detta paket utvecklade jag själv. En del klasser kunde jag återanvända från javapaketet ir vilket utvecklades av Ray Mooney vid University of Texas at Austin. Nedan återges en tabell på de viktigaste javaklasser som jag använde vid mina experiment:

Klass	*Beskrivning*	*Författare*	*Paket*
(abstract) Document	Stöd för borttagning av stoppord och bag-of-words representation.	Mooney	ir.vsr
DocumentIterator	För att iterera över en kollektion med dokument och skapa Document objekt av dessa.	Plenk	ir.erp
Example	Representera tränings- och testexempel för maskininlärning på text.	Plenk	ir.erp
HashMapVector	En datastruktur för termvektorer där dokument lagras som en HashMap vilken mappar termer och vikter.	Mooney	ir.vsr
InvertedIndex	För att vikta termer, reducera dimensioner och skapa ett inverterat index.	Plenk	ir.erp
KNN	Maskininlärningsmetoden k-Nearest Neighbor.	Plenk	ir.erp
Measures	Utvärderingsmått och kvantiteter för multipelt binära textklassificeringsuppgifter.	Plenk	ir.erp
Porter	Omvandling av ord till grundform.	Lazarinis	ir.utilities
Reuters21578	Representerar dokument från kollektionen Reuters-21578. Genomför lexikalisk analys.	Plenk	ir.erp.vsr
Reuters21578Handler	En SAX parser som läser xml-dokument och extraherar ut text.	Plenk	ir.erp.vsr

Tabell 1. Klasser som användes vid utveckling av automatisk textklassificerare.

För att ladda ner paketet som jag sammanställde så finns det tillgängligt vid adressen:
http://webb.informatik.gu.se/~s97plenk/atc/

För att ladda ner ir-paketet som utvecklades av Mooney finns det tillgängligt vid adressen:
http://www.cs.utexas.edu/users/mooney/ir-course/

För att kunna utvärdera framtagen textklassificerare, använde jag kollektionen Reuters-21578. Kollektionen består av 21,578 dokument vilka var publicerade vid Reuters nyhetsbyrå under året 1987. Kollektionen används vid många experiment inom textklassificering, och fördelen är att man objektivt kan jämföra olika resultat med varandra. Närmare beskrivning av kollektionen ges i Appendix 1 – Reuters-21578, och för att ladda ner den finns den tillgänglig vid adressen:
http://www.daviddlewis.com/resources/testcollections/reuters21578/

Under prototyparbetet med ERP Text Miner utvecklades en serie agenter. För att kunna implementera grundläggande egenskaper hos dessa använde jag mig av paketet CIAgent. CIAgent är ett javapaket som inkapslar en abstrakt design för lösningar relaterade till autonoma agenter. Paketet utvecklades Joseph och Jennifer Bigus och finns tillgänglig på den cd-skiva som ingår i deras bok[7].

2.5 Analys

För att skapa överskådlighet i mängden insamlad data analyserades detta på ett sätt som kan beskrivas med Miles & Hubermans flödesmodell. Miles & Huberman beskriver dataanalysen som fyra parallella flöden: datainsamling, datareduktion, dataåtergivning, och dragande av slutsatser. De olika aktiviteterna är sammanflätade och utförs så länge som arbetet pågår[8].

Datainsamlingen var den aktivitet där jag valde ut material, och datareduktionen var den aktivitet där materialet abstraherades, fokuserades och överflyttades från original källorna. Dessa aktiviteter pågick under hela arbetet.

Dataåtergivning var en annan betydande aktivitet för att skapa överskådligt under analysen. Under dataåtergivningen presenterade och organiserade jag informationen, vilket på så sätt gjorde det enklare att dra slutsatser. När jag drog vissa slutsatser gav detta mer precist fokus och stimulerade på så sätt till ny datainsamling och datareduktion.

2.6 Validitet och reliabilitet

För att det skall vara möjligt verifiera rimliga slutsatser bör materialets validitet och reliabilitet testas[9]. *Validitet* innebär att man verkligen har undersökt det man ville undersöka och ingenting annat. Undersökningen får alltså inte missa några viktiga aspekter av det som står under fokus. *Reliabilitet* är ett mått på korrektheten hos ett arbete. En hög reliabilitet innebär att mätningar och undersökningar är korrekt gjorda. När ett arbete har hög reliabilitet kommer, till exempel, två skilda undersökningar med samma syfte att under samma period och samma population att ge samma resultat.

Den stora mängd material som ligger till grund för arbetet samt allt prototyparbete, har tjänat till att bredda fokus för att på så sätt inte missa några viktiga aspekter. Därför stödjer både använt material och allt utvecklingsarbete en god validitet. För att säkerställa reliabiliteten har arbetet grundats i vedertagna teorier och modeller. Experiment har utförts på standardiserade sätt och korrektheten hos prototypen har kontinuerligt utvärderats.

[7] Bigus & Bigus: “Constructing Intelligent Agents Using Java”
[8] Miles & Huberman: “Quality data analysis”
[9] Svenning: ”Metodboken”

3 AUTOMATISK TEXTKLASSIFICERING

3.1 Inledning

Detta kapitel beskriver centrala begrepp, utvecklingssteg, algoritmer, och utvärderingsmetoder inom automatisk textklassificering. Kapitlet syftar till att översiktligt redogöra för bakomliggande teori som väglett framställningen av ERP Text Miner. Läsare som önskar mer detaljer, och flera perspektiv inom området, hänvisas med fördel till källorna: Aas & Eikvil[10], Sebastiani[11], samt Baeza-Yates och Ribeiro-Neto[12].

3.2 Definition av klassificeringsuppgiften

Sebastiani[13], definierar *textklassificering* (eller *dokumentklassificering*), som uppgiften att göra en *tilldelning* av ett värde från {0,1} till varje ingång i *beslutsmatrisen*:

	d_1	...	...	d_j	...	...	d_n
c_1	a_{11}	...	...	a_{1j}	...	...	a_{1n}
...	...	...	...	...	...	...	...
c_i	a_{i1}	...	...	a_{ij}	...	...	a_{in}
...	...	...	...	...	...	...	...
c_m	a_{m1}	...	...	a_{mj}	...	...	a_{mn}

där $C = \{c_1,...,c_m\}$ är en uppsättning på förhand definierade *klasser*, och $D = \{d_1,...,d_n\}$ är en uppsättning dokument att klassificera. Ett värde av 1 för a_{ij} tolkas som beslutet att arkivera d_j under c_i medan värdet 0 tolkas som beslutet att inte arkivera d_j under c_i.

Fundamentalt för att förstå denna uppgift är två observationer:

- klasserna är endast symboliska etiketter. Ingen ytterligare kunskap om deras ”innebörd” finns tillgänglig vid processen att bygga en klassificerare;
- de termer som representerar dokumentet, väljs ut baserat på dokumentets innehåll, och inte på basis av någon metadata[14].

[10] Aas, K, & Eikvil, L.: “Text Categorisation: a survey”
[11] Sebastiani, F: “A tutorial on automated text categorisation”
[12] Baeza-Yates, & Ribeiro-Neto: “Modern Information Retrieval”
[13] Sebastiani, F: “A tutorial on automated text categorisation”
[14] Ibid.

3.3 Användningsområden

Automatisk textklassificering går tillbaka till det tidiga 60-talet. Och sedan dess har metoden tillämpats i flera sammanhang. För att ytterligare belysa textklassificering, ges här en kort översikt över några.

3.3.1 Automatisk indexering

Det första användningsområdet, var i samband med automatisk dokumentindexering för IR-system (information retrieval systems). I dessa system tilldelas varje dokument en eller flera nyckelord, eller nyckelordsfraser, som beskriver dokumentets innehåll. Då detta tidigare gjordes manuellt, innebar en automatisering av processen stora kostnadsbesparningar. Exempel på system för automatisk indexering ges i litteraturen som till exempel AIR/PHYS[15] och AIR/X[16].

3.3.2 Organisation av dokument

Övergripande, i alla sammanhang där behovet finns att organisera dokument, kan detta med fördel göras automatiskt. Ett typexempel är behovet att klassificera annonser vid en tidning. Vid annonsredaktionen inkommer varje dag en mängd annonser som skall publiceras under olika kategorier. Kategorier kan till exempel vara bilar, båtar, datorer, …, etc. Då de flesta redaktioner gör denna sortering för hand, och man varje dag måste hantera en betydande mängd, kan detta med fördel göras med automatiska tekniker.

3.3.3 Filtrering av dokument

Filtrering av dokument är aktiviteten att klassificera en *dynamisk*, snarare än en statisk, uppsättning dokument, där det råder en kontinuerlig ström av inkommande dokument. Filtrering är ett specialfall av klassificering, där klassificeringen görs i två klasser, relevant och irrelevant[17]. Ett typexempel är fallet med produktion av nyheter, där informationsproducenten är en nyhetsbyrå (till exempel Reuters) och informationskonsumenten är en tidning. I detta fall, syftar filtreringssystemet till att blockera ut de dokument som konsumenten inte är intresserad av (till exempel alla nyheter som inte handlar om sport, i fallet en sporttidning). Konstruktion av filtreringssystem med maskininlärande tekniker diskuteras frekvent i litteraturen. Till exempel, Shütze, Hull och Pedersen jämför olika maskininlärande metoder för routing[18].

3.3.4 Katalogsökning

Automatisk klassificering har också fått mycket uppmärksamhet i samband med Internet.
Ett typexempel är att organisera Webbsidor i kataloghierarkier, och på så sätt erbjuda användarna ett betydligt enklare sätt att leta efter dokument. Portalen Yahoo[19] erbjuder till exempel en sådan tjänst.

[15] Biebricher, Fuhr, Lustig, & Schwantner: “The automatic indexing system AIR/PHYS. From research to application.”
[16] Fuhr, Hartman, Knorz, Lustig, Schwantner, & Tzeras: “AIR/X – a rule-based multistage indexing system for large subject fields”
[17] Sebastiani, F: “A tutorial on automated text categorisation”
[18] Shütze, Hull, & Pedersen: “A comparison of classifiers and document representations for the routing problem”
[19] www.yahoo.com

3.4 Maskininlärning för textklassificering

Tidigare under ’80-talet byggdes klassificeringssystem manuellt[20]. Man konstruerade då expertsystem med regler av typen **if**(formula) **then**(klass). Funktionen var om dokument uppfyllde (formula) så klassificerades det under (klass). Dessa expertsystem var oftast mycket effektiva, men samtidigt mycket dyra att bygga och underhålla. Förklaringen var den så kallade kunskaps-förvärvs-flaskhalsen (*knowledge-acquisition-bottleneck*), vilken innebär att stora resurser måste allokeras vid konstruktion och underhåll av systemet.

Sedan början av ’90-talet har *maskininlärning* (*machine learning*) blivit den dominerande metoden att konstruera automatiska klassificeringssystem[21]. Vid maskininlärning inducerar ett program automatiskt en klassificerare för kategorin c_i. Detta görs genom att programmet ”observerar” karaktäristiken hos en uppsättning dokument som tidigare manuellt har klassificerats av en expert. Utifrån denna inducerade kunskap kan programmet sedan bedöma om ett nytt dokument skall klassificeras under c_i, eller inte.

3.4.1 Träning och testning

Maskininlärning utgår att det existerar ett *initialt korpus* (d v s en på förhand given textmängd) $Co = \{\overline{d_1},...,\overline{d_S}\}$ av dokument som tidigare har klassificerats under en uppsättning klasser $C = \{c_1,...,c_m\}$ varvid inom klassificerarn måste operera[22]. Detta betyder att det finns ett korpus med en *korrekt beslutsmatris*:

	Träningsdel				Testningsdel			
	$\bar{d}_1$	...	...	$\bar{d}_g$	$\bar{d}_{g+1}$	...	...	$\bar{d}_s$
c_1	ca_{11}	...	...	ca_{1g}	$ca_{1(g+1)}$	...	...	ca_{1s}
...	...	...	...	...	...	...	...	...
c_i	ca_{i1}	...	...	ca_{ig}	$ca_{i(g+1)}$	...	...	ca_{is}
...	...	...	...	...	...	...		...
c_m	ca_{m1}	...	...	ca_{mg}	$ca_{m(g+1)}$	...	...	ca_{ms}

Värdet 1 för ca_{ij} tolkas som en indikation från experten att arkivera d_j under c_i, medan värdet 0 tolkas som en indikation från experten att inte arkivera d_j under c_i. Ett dokument d_j refereras oftast som ett *positivt exempel* (*positive example*) för c_i om $ca_{ij} = 1$, ett *negativt exempel* (*negative example*) för c_i om $ca_{ij} = 0$.

För att kunna utvärdera klassificerarns klassificeringsförmåga, så delas textsamlingen typiskt upp i två uppsättningar:

[20] Sebastiani, F: “A tutorial on automated text categorisation”
[21] Ibid.
[22] Ibid.

- en uppsättning för *träning* $Tr = \{\bar{d}_1,...,\bar{d}_g\}$. Från denna uppsättning med exempeldokument hämtas den karaktäristik varvid med klassificerarna induceras (framkallas);

- en uppsättning för *testning* $Te = \{\bar{d}_{g+1},...,\bar{d}_s\}$. Denna uppsättning används med syfte att utvärdera prestanda hos den framtagna klassificeraren. Varje dokument i *Te* kommer att matas in till klassificeraren, och klassificerarens arkiveringsbeslut jämförs då med det beslutet som fattades av experten; ett mått för prestanda baseras på hur ofta värden för a_{ij} erhållna av klassificeraren stämmer överens med värdena för ca_{ij} givna av experten.

3.5 Extrahering av termer

3.5.1 Förbehandling

Först fasen i textklassificering handlar om att förebehandla dokumenten, vilka typiskt består av strängar av tecken, till en form lämplig för den maskininlärningsalgoritm man sedan väljer. Typiskt här, är att man tar fram nyckelord, eller *termer*, som man sedan kan indexera på, s.k. *indextermer*.

Inte alla ord i dokument är lika signifikanta att representera semantiken hos ett dokument. I skrivet språk, bär en del ord mer *innebörd* än andra. Vanligtvis är substantiv den ordklass som mest representerar innehållet i ett dokument[23]. Därför är det oftast mödan värt att förebehandla texten i dokumentkollektionen för att avgör vilka ord, eller termer, som skall används som indextermer.

Förbehandling av dokument är en procedur som kan delas upp i fyra textopererande steg (eller transformationer)[24]:

- lexikalisk analys (att identifiera själva orden i dokumenten);
- borttagning av märkord och andra meta-ord (t.ex. HTML, SGML, XML etc.);
- borttagning av stoppord (stopwords) d.v.s. frekvent förekommande informationsfattiga ord;
- framtagning av ordens stjälk (wordstemming), d.v.s. ta fram ordens grundform.

Lexikalisk analys är processen att omvandla en ström av tecken (texten i dokumenten) till en ström av ord (ord som kandiderar till att senare användas som indextermer). Således är det huvudsakliga målet under den lexikaliska analysen att identifiera orden i texten[25]. Vid en första anblick verkar detta vara trivialt, allt som tycks behöva göras är att identifiera de tecken som avskiljer orden från varandra t.ex. blankstegstecken. Men det behövs göras mer. Speciellt behöver man ta hänsyn till fyra följande fall: siffror, bindesträck, punkteringstecken, samt versaler och gemener (stora och små bokstäver)[26]. Vad gäller s*iffror* är de ofta inga bra indextermer – utan omgivande ord är de vaga. Således tas dessa oftast bort. Ett annat beslut berör de ord som är hopbundna med *bindesträck*. Här brukar man välja en generell regel som tillämpas på alla sådana ord. Till exempel ordet ”state-of-the-art” kan då brytas ner till tre ord: ”state”, ”of”, ”the”, och ”art”. Beträffande *punkteringstecken* tas dessa normalt oftast bort, till exempel förkortningen ”t.ex.” bryts då ner till ”t” och ”ex”. Beträffande *versaler och gemener*, väljer man ofta omvandla alla tecken till antingen stora eller små bokstäver.

[23] Baeza-Yates, & Ribeiro-Neto: “Modern Information Retrieval”
[24] Aas, K, & Eikvil, L.: “Text Categorisation: a survey”
[25] Ibid.
[26] Baeza-Yates, & Ribeiro-Neto: “Modern Information Retrieval”

Borttagning av märkord är processen att ta bort märkinformation eller annan meta information från dokumenten[27]. Märktaggar och annan metainformation är inte bra som indextermer då de i sig själva inte representerar själva innehållet i dokumentet. Dessa tas då bort. Exempel på märkspråk är HTML, SGML, XML, m.fl.

Borttagning av stoppord (stopwords) är processen filtrera bort frekvent förekommande informationsfattiga ord[28]. Inte alla ord är bra för att representera semantiken i dokument. Faktum är, att de ord som förekommer i 80% eller mer bland dokumenten är värdelösa att indexera med. Anledning är deras låga diskrimineringsvärde, d.v.s. de har låg urskiljningskraft. Sådana frekvent förekommande ord kallas för *stoppord* (*stopwords*), och dessa önskar man alltså rensa bort[29]. Exempel på stoppord är prepositioner, konjunktioner, pronomen m.fl. Stoppord är givetvis språkberoende och därför har man olika listor för det språk som gäller. Borttagning av stoppord har ytterligare en stor fördel. Det reducerar storleken på indexstrukturen betydligt. Faktum är, att man oftast kan komprimera indexstrukturen med 40% eller mer endast genom att eliminera stoppord.

Ta fram ordens stjälk, (wordstemming), är processen att omvandla ord till deras grundform, d.v.s. ta bort ordens affix – prefix och/eller suffix[30]. En *stjälk*, (*stem*), är en delmängd av ett ord som återstår efter man har tagit bort dess affix (d.v.s. prefix/suffix). Ett typexempel på en stjälk är ordet *dator* vilket är en stjälk för varianterna: *datorn*, *datorer*, *datorerna*. Stjälkar anses användbart för att öka systemets klassificeringsförmåga, därför att de reducerar varianter av samma rot-ord för ett allmänt koncept. Ytterligare fördelar med ordstjälkning är att storleken på indexstrukturen minskar betydligt därför att antalet distinkta indextermer reduceras[31]. Vid ordstjälkning är det viktigast att ta bort suffixdelen, därför de flesta varianter av ord uppstår genom att tillsätta suffix. Det finns flera kända algoritmer för suffixborttagning, och den mest populära för engelsk text är Porter Stemmer[32], på grund av dess enkelhet och elegans. Trots att Porters Stemmer är relativt enkel genererar den mycket goda resultat, klart jämförbara med mer komplicerade algoritmer. Arbete med svenska stemmingsalgoritmer görs också och har visat sig förbättra resultaten i textbehandlingssammanhang[33].

3.5.2 Representation

Den andra fasen vid textklassificering handlar om att *representera*, de förebehandla dokumenten och deras termer (nyckelord) på ett lämplig sätt. Då textklassificering med maskininlärande metoder bygger på matematiska modeller, önskar man en representationsform där det går att uttrycka dokument och termer matematiskt. Vanligt är att man använder den statistiska modellen *vektor-rymd-modellen*[34] (*vector-space-model*) där dokumentets termer *viktas* och motsvarar en vektor i en rymd[35] (kallas också för *bag-of-words* representationen för dokument). Dessa vektorer inordnas sedan i en s.k. *term-dokument-matris* (*term-document-matrix*).

[27] Aas, K, & Eikvil, L.: "Text Categorisation: a survey"
[28] Ibid.
[29] Baeza-Yates, & Ribeiro-Neto: "Modern Information Retrieval"
[30] Aas, K, & Eikvil, L.: "Text Categorisation: a survey"
[31] Baeza-Yates, & Ribeiro-Neto: "Modern Information Retrieval"
[32] Porter: "An Algorithm for Suffix Stripping"
[33] Carlberger, Dalianis, Hassel & Knutsson: "Improving precision in information retreival for Swedish using stemming"
[34] Aas, K, & Eikvil, L.: "Text Categorisation: a survey"
[35] Baeza-Yates, & Ribeiro-Neto: "Modern Information Retrieval"

Vektor-rymd-modellen utgår från att det existerar *t* distinkta termer (nyckelord) efter förbehandlingen. Dessa termer kallas för *indextermer* eller för *vokabulären*, de formar då en vektorrymd där $Dimension = t = |vokabulären|$. Varje term, *i*, i dokumentet, *j*, ges en vikt, w_{ij}. Dokumenten kan då uttryckas som *t-dimensionella vektorer*: $d_j = (w_{1j}, w_{2j}, ..., w_{tj})$.

Term-dokument-matrisen inordnar sedan dessa t-dimensionella vektorer i en matris (se tabell 2). Varje kolumn motsvarar en term (nyckelord), varje rad motsvarar ett dokument, och varje matrisingång motsvarar ”vikten” hos termen i dokumentet; noll innebär att termen saknar signifikans eller helt enkelt inte existerar i dokumentet.

	T_1	T_2	...	T_t
D_1	w_{11}	w_{21}	...	w_{t1}
D_2	w_{12}	w_{22}	...	w_{t2}
...	...	...	...	…
...	...	...	...	…
D_n	w_{1n}	w_{2n}	...	w_{tn}

Tabell 2. Term-dokument-matrisen.

I med att varje term (nyckelord) inte normalt förekommer i varje dokument, så brukar TD-matrisen normalt bli gles (eng. sparse)[36]. Antalet termer totalt i dokumentkollektionen kan däremot bli mycket stort. Detta kan senare ställa till problem vid textklassificering. I avsnitt 3.5.4 beskrivs några sätt att minska antalet dimensioner TD-matrisen.

TD-matrisen är en mycket populär representationsform. Dess generella format möjliggör att användningen av många olika maskininlärningsmetoder.

3.5.3 Viktning

Nästa steg är vikta termerna. Det finns olika sätt att beräkna vikten, w_{ik} , för termen, *i*, i dokumentet, *k*, men de flesta sätt bygger på två empiriska observationer rörande text:

- Ju fler gånger en term (ord) förekommer i ett dokument, ju mer relevant är det rörande ämnet i dokumentet.

- Ju fler gånger en term (ord) förekommer i alla dokument i dokumentkollektionen, sämre diskrimineringskraft (urskiljningsförmåga) har det mellan dokumenten[37].

Som utgångspunkt, låt f_{ik} vara frekvensen av termen, *i*, i dokumentet *k*, och här följer några exempel på vikningsprinciper som utgår från denna kvantitet.

[36] Ibid.
[37] Aas, K, & Eikvil, L.: “Text Categorisation: a survey”

Boolesk-viktning
Det enklaste sättet är boolesk-viktning[38], där vikten är 1 om termen förekommer i dokumentet och 0 om det inte gör det:

$$w_{ik} = \begin{cases} 1 \quad om \quad f_{ik} > 0 \\ 0 \quad annars \end{cases}$$

Termfrekvens-viktning
Termfrekvens-viktning[39] är ett annat enkelt tillvägagångssätt, där frekvensen av termen i dokumentet avgör vikten:

$$w_{ik} = f_{ik}$$

tfx**idf-viktning**
De tidigare två principer tar inte hänsyn till frekvensen av termen genom alla dokument i kollektionen. En välkänt tillväggångssätt att beräkna termens vikt är med *TFxIDF*-viktning. TFxIDF sätter vikten till termen i, i dokumentet k, i proportion till antalet förekomster av termen i dokumentet, och en inverterad proportion av antalet dokument i kollektionen där termen förekommer minst en gång[40].

Låt f_{ik} fortvarande vara frekvensen av termen, i, i dokumentet k. I med att man ofta önskar normalisera *termfrekvens* (*tf*) över hela kollektionen så låt $tf_{ik} = f_{ik} / \max\{f_{ik}\}$. Låt vidare, idf_i vara den inverterade dokumentfrekvensen av termen i, vara lika med $\log(N / df_i)$, där N är det totala antalet dokument, och df_i är dokumentfrekvensen av termen i, och där $\log$ används för att dämpa effekten relativt *tf*.

Sår är vikten, w_{ik} , för termen, i, i dokumentet, k:

$$w_{ik} = tf_{ik} idf_i = tf_{ik} \log(N / df_i)$$

Via tfxidf viktning ges en term som frekvent förekommer i dokumentet, men övrigt är sällsynt i dokumentkollektionen, en hög vikt. Många andra förslag att beräkna termvikter har föreslagits (ytterligare en metod återges nedan). Experimentellt har, tfxidf, visats fungera bra[41].

ltc-viktning
Vid tfxidf-viktning tar man ingen hänsyn till att dokument kan vara av olika längd. Långa och ordrika dokument återupprepar oftast samma term flera gångar. Långa dokument innehåller också ett större antal olika termer. Resultat blir att man får en hög termfrekvensfaktor för långa dokument, det blir alltså en högre chans att matcha långa dokument framför kortare dokument. För att kompensera denna effekt, brukar man *normalisera* termvikterna. Normalisering är ett sätt att införa ett straff för

[38] Ibid.
[39] Ibid.
[40] Baeza-Yates, & Ribeiro-Neto: “Modern Information Retrieval”
[41] Ibid.

termvikter i långa dokument. *Cosinusnormalisering* är en effektiv normaliseringsteknik. Varje termvikt i dokumentet divideras med den Euklidiska längden hos en *tfxidf-viktad* dokumentvektor, $\sqrt{w_1^2 + w_2^2 + ... + w_n^2}$, där w_i är *tfxidf-vikten* för termen *i* i dokumentet. Detta ger vikten för termen:

$$\frac{tfxidf\_vikt}{Euklidiska\_längden\_på\_dokumentvektorn}$$

Vilket återges med följande formula[42]:

$$w_{ik} = \frac{\log(tf_{ik} + 1.0) * \log(N / df_i)}{\sum_{j=1}^{M} \left[\log(tf_{jk} + 1.0) * \log(N / df_j)\right]^2}$$

Där logaritmen för termfrekvensen används istället för enbart termfrekvensen, vilket reducerar den effekt som uppstår vid stora frekvensskillnader.

3.5.4 Reducering av dimensioner

Ett centralt problem vid textklassificering med statistiska metoder, är det höga antalet dimensioner. Det finns en dimension för varje unik term i dokumentkollektionen, vilket kan bli hundratals eller tusentals. De flesta klassificeringstekniker kan inte hantera en så stor uppsättning, då detta blir extremt beräkningsintensivt, och att resultatet blir osäkert på grund av avsaknaden av tillräcklig träningsdata. Därför finns behovet att reducera antalet dimensioner[43]. De flesta dimensionsreducerande tekniker kan klassificeras i en av två kategorier; feature selection eller reparameterisering.

Ibland genomförs en s.k. *reparameterisering*, vilket innebär att man transformerar term-dokument-matrisen till en rymd med färre dimensioner. *Latent Semantic Indexing*[44] är ett exempel på en sådan metod.

När reparameterisering är olämplig kan istället en rensning, *feature selection*, av term-dokument-matrisen genomföras. Vid feature selection försöker man avlägsna de ord från dokumenten som har lågt informationsvärde. Yang och Pedersen[45], har genomfört en utvärdering av fem metoder för feature selection; Document Frequency Thresholding, Information Gain, X^2, Mutual Information, och Term Strength. Vid deras experiment drogs slutsatsen att de tre förstnämnda genererade bäst resultat.

I detta arbete har dessa fem metoder studerats, och nedan beskrivs de översiktligt.

[42] Buckley, Salton, Allan & Singhal: "Automatic Query Expansion Using SMART: TREC 3"
[43] Aas, K, & Eikvil, L.: "Text Categorisation: a survey"
[44] Deerwester, Dumais, Furnas, Landauer, & Harshman: "Indexing by Latent Semantic Analysis"
[45] Yang & Pedersen: "Feature selection in statistical learning of text categorization"

Document Frequency Thresholding (DFT)
Dokumentfrekvensen för ett ord är antalet dokument där ordet förekommer. Vid DFT beräknas först dokumentfrekvensen för varje ord bland träningsdokumenten. Därefter avlägsnas de ord vars dokumentfrekvens är under ett givet tröskelvärde.

Vid DFT utgår man från att ovanliga ord har lågt informationsvärde för klassificeringsbeslut och därför kan dessa avlägsnas. Eller med andra ord, de mest värdefulla orden för klassificeringsbeslut är de som frekvent förekommer i kollektionen.

DFT är överraskande effektiv för att öka klassificeringsprestanda. DFT har systematiskt utvärderats av Yang et al[46], och användes först av Apté et al[47].

Information Gain (IG)
Vid IG mäter man antalet bitar av information som fås inför ett klassificeringsbeslut, genom att känna till närvaron eller frånvaron av ett ord i ett dokument[48].

Låt $c_1,\ldots,c_k$ stå för möjliga kategorier. IG för ordet *w*, beräknas då genom:

$$\begin{aligned} IG(w) = & -\sum_{j=1}^{K} P(c_j)\log P(c_j) \\ & + P(w)\sum_{j=1}^{K} P(c_j \mid w)\log P(c_j \mid w) \\ & + P(\bar{w})\sum_{j=1}^{K} P(c_j \mid \bar{w})\log P(c_j \mid \bar{w}) \end{aligned}$$

Här uppskattas $P(c_j)$ från andelen dokument i dokumentkollektionen som tillhör klassen c_j, och $P(w)$ från andelen dokument där ordet *w* förekommer. $P(c_j \mid w)$ beräknas från andelen dokument från klassen c_j som minst har en förekomst av ordet *w*, och $P(c_j \mid \bar{w})$ som andelen av dokument från klassen c_j som inte innehåller ordet *w*.

IG beräknas för varje ord i träningsdelen, och de ord vars IG-värde är under ett givet tröskelvärde tas sedan bort.

[46] Ibid.
[47] Apté, Damerau & Weiss: ”Automated learning of decisions rules for text categorization”
[48] Yang & Pedersen: ”Feature selection in statistical learning of text categorization”

X2

Vid X^2 mäts saknaden av oberoende mellan ordet *w* och klassen c_j[49]. Följande formula används:

$$X^2(w,c_j) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

Här står *A* för antalet dokument från klassen c_j som innehåller ordet *w*, *B* är antalet dokument som innehåller *w* men som inte tillhör klassen c_j, *C* är antalet dokument från klassen c_j som inte innehåller ordet *w*, *D* är antalet dokument som inte tillhör klassen c_j samt inte innehåller ordet *w*, och *N* är det totala antalet dokument.

X^2 har ett naturligt värde noll om ordet *w* och klassen c_j är oberoende. För varje kategori beräknas X^2 mellan varje unikt ord och given kategori. Två olika mått kan sedan beräknas:

$$X^2(w) = \sum_{j=1}^{K} P(c_j) X^2(w,c_j)$$

eller

$$X^2_{\max}(w) = \max_j X^2(w,c_j)$$

Mutual Information (MI)

MI[50] är ett vanligt kriterium vid statistiska modeller. Om man föreställer sig en fyrfältstabell som beskriver utfallet av två alternativ rörande ordet *w* och kategorin c_j, så står *A* för antalet gånger ordet *w* och kategorin c_j saminträffar, *B* är antalet gånger *w* inträffar men inte c_j, *C* är antalet gånger c_j inträffar men inte *w*, och *N* är det totala antalet dokument. MI mellan ordet *w* och kategorin c_j uppskattas sedan med:

$$MI(w,c_j) \approx \log \frac{A \times N}{(A+C) \times (A+B)}$$

MI har ett naturligt värde noll om ordet *w* och kategorin c_j är oberoende. För att sedan uppskatta vilka ord som skall väljas ut kan två alternativa mått användas:

$$MI_{avg}(w) = \sum_{j=1}^{K} P(c_j) MI(w,c_j)$$

och

[49] Ibid.
50 Ibid.

$$MI_{\max}(w) = \max_{j=1}^{K}\{MI(w, c_j)\}$$

Term Strength (TS)
TS är ett mått som först tillämpades för textklassificering av Yang och Wilbur[51]. TS uppskattar ordets betydelse, eller styrka, utifrån hur vanligt förekommande det är i "nära-relaterade" dokument[52].

Metoden använder träningsdokumenten till att identifiera dokumentpar vars likhet (som mäts med kosinuslikhet) är över ett givet tröskelvärde. Sedan beräknas TS utifrån den villkorliga sannolikheten att ett ord förekommer i den andra halvan av ett par relaterade dokument, givet att det förekommer i första halvan.

Låt *x* och *y* vara ett godtyckligt par distinkta men relaterade dokument, och *w* vara ett ord, då beräknas styrkan hos ordet genom:

$$s(w) = P(w \in y \mid w \in x)$$

TS kriteriet skiljer sig radikalt från ovan nämnda metoder. Det baseras på klustring av dokument. Man antar att dokument med många gemensamma ord är relaterade, och att ord i relaterade dokument är relativt informativa för klassificeringsbeslut.

3.6 Maskininlärningsmetoder

Det finns många olika maskininlärningsmetoder (algoritmer) för textklassificering som har utvecklats och utvärderats under årens lopp. Exempel är Support Vector Machines, k-Nearest Neighbor, naive Bayes, Decision Trees, Neurala Nätverk, med många andra. I detta avsnitt presenterar jag två mycket populära sådana metoder: k-Nerest Neighbor och naive Bayes. Anledningen till att jag väljer att presentera just dessa två, är att dels redovisar de goda resultat i litteraturen, dels har de här i detta arbete, både teoretiskt och praktiskt testats samt utvärderats. För de läsare som önskar mer information om andra algoritmer, hänvisas med fördel till Aas et. al[53]., Herou[54] och Sebastiani[55].

Den presentation som görs här, är i huvudsak kvalitativ än kvantitativ, d.v.s. fokus är på metoden att inducera klassificeraren, snarare än för den prestanda som kan få ut ur den.

3.6.1 k-Nearest Neighbor

k-Nearest Neighbor (kNN) är en metod som går ut på att hitta dokument i träningsdelen som liknar det dokument som skall klassificeras. För att besluta om d_j skall klassificeras under c_i, så tittar kNN efter huruvida de *k* träningsdokument (grannarna) mest lika d_j också har klassificeras under c_i; om

51 Yang, & Wilbur: “Using corpus statistics to remove redundant words in text categorization”
[52] Yang & Pedersen: ”Feature selection in statistical learning of text categorization”
[53] Ibid.
[54] Herou: ”Ett träningsbart verktyg för att klassificera nyhetstexter”
[55] Sebastiani, F: “A tutorial on automated text categorisation”

svaret är positivt för en tillräckligt stor andel, fattas ett positivt klassificeringsbeslut, annars görs ett negativt beslut[56].

Att endast använda den närmaste (den mesta lika) träningsvektorn (grannen) för att avgöra klasstillhörighet är föremål för felaktigheter då:

- det kan finnas ett icke-typiskt exempel;
- det kan finnas fel i klassmarkeringen hos ett ensamstående träningsdokument.

Ett mer robust alternativ är då att hitta de *k* mest liknande träningsvektorerna och returnera majoritetsklassen hos dess *k* exempel. Värdet på *k* är typiskt ojämnt för att undvika olyckliga bindningarna mellan klasser. 3 och 5 är mycket vanliga värden för *k*.

För att avgöra likheten mellan dokumentvektorer använder sig kNN av ett likhetsmått (eller distansmått). Ett *likhetsmått* är en funktion som beräknar *graden av likhet* mellan två vektorer. Genom att använda ett likhetsmått mellan ett dokument och dess granndokument är det möjligt att:

- ranka de *k* närmsta grannarna efter relevans/klasstillhörighet.
- tilltvinga ett speciellt *tröskelvärde* så att ett positivt eller negativt klassificeringsbeslut kan fattas

Enklast är att använda sig av Euklidiskdistans, men vid textklassificering är *cosinuslikhet* hos TFxIDF viktade vektorer typiskt mest effektivt.

Cosinuslikhet mäter cosinus av vinkeln mellan två vektorer.

$$CosSim(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$

Där w_{ij} är vikten hos termen *i*, i dokumentet *j*, och w_{iq} är vikten hos termen *i*, i det dokument som jämförs.

Träningsfasen hos kNN är att representera alla träningsdokument som t-dimensionella vektorer och att inordna dessa i en term-dokument-matris. Att sedan *testa* ett dokument innebär att:

- omvandla testdokumentet till en t-dimensionell vektor;
- beräkna likheten mellan denna vektor och alla vektorer i term-dokument-matrisen;
- tilldela testdokumentet klassen hos de mest liknande exempel i matrisen (om ett vist tröskelvärde överskrids).

kNN är en s.k. långsamt lärande (lazy learning) exempelbaserad metod, därför det inte finns någon direkt träningsfas. Träningsfasen är istället att representera träningsdokumenten i en term-dokument-matris. Den största arbetet är likhetsberäkningen mellan träningsvektorerna och testvektorn när de *k* närmsta grannarna skall hittas. När ett inverterad index används för träningsdokumenten, är tidskomplexiteten O(L * N/M) där L, är antalet element hos de dokumentvektorer som är större än noll, M, är längden hos dokumentvektorn och N, är antalet träningsexempel.

[56] Ibid.

3.6.2 Naive Bayes

Naive Bayes, bygger på Bayes teorem för betingad sannolikhet. Genom att fastställa förekomstfrekvensen hos varje enskilt ord sammanfaller med ett dokuments klasstilldelning, kan sannolikheten för ett dokuments klasstillhörighet beräknas[57].

För att med hjälp av Bayes teorem kunna beräkna sannolikheten för ett visst dokuments klasstillhörighet, behövs ett antal sannolikheter beräknas. $P(c_j \mid d)$ är sannolikheten att ett dokumentet skall tillhöra klassen c_j, givet ordförekomsterna i dokumentet som en vektor *d*. $P(c_j)$ är sannolikheten för att något dokument skall tillhöra klassen c_j, $P(d \mid c_j)$ är sannolikheten att ordförekomsten *d*, skall förekomma i klassen c_j, och $P(d)$ är sannolikheten att ordförekomsten *d*, alls skall förekomma. Bayes regel gäller då:

$$P(c_j \mid d) = \frac{P(c_j)P(d \mid c_j)}{P(d)}$$

Att metoden kallas för *naiv*, beror på den förenkling som möjliggör att slutföra beräkningen. Man utgår nämligen att förekomsten av varje ord är oberoende av de andra ordens förekomst. Man gör därmed den naiva beräkningen, att sannolikheten för en klasstillhörighet c_j, givet ordvektorn *d*, är:

$$P(c_j \mid d) = \frac{P(c_j)\prod_{i=1}^{M} P(d_i \mid c_j)}{P(d)}$$

$P(d_i \mid c_j)$ är sannolikheten för att dokumentets värde på d_i skall förekomma i något dokument i klassen c_j. Om syftet är att välja den mest sannolika klassen för ett dokument, blir nämnaren densamma för alla klasser c_j som skall övervägas för dokumentet, eftersom det hela tiden gäller samma dokument. Nämnaren kan därmed tas bort och förenklingen blir:

$$P(c_j \mid d) = P(c_j)\prod_{i=1}^{M} P(d_i \mid c_j)$$

där M är antalet ord (eller termer) i vektorn *d*. Dock skall noteras att $P(c_j \mid d)$ inte längre kan betraktas som en sannolikhet eftersom nämnaren tagits bort och resultatet därmed skalats om med en okänd faktor $P(d)$. Det återstår att ta fram $P(d \mid c_j)$, d.v.s. sannolikheten för att respektive ord skall förekomma givet en klass c_j, samt $P(c_j)$, d.v.s. sannolikheten för att klassen skall tilldelas något dokument. Dessa kan uppskattas enligt:

[57] Aas, K, & Eikvil, L.: "Text Categorisation: a survey"

$$P(d_i \mid c_j) \approx \frac{1 + N_{ij}}{M + \sum_{k=1}^{M} N_{kj}}$$

$$P(c_j) \approx \frac{N_j}{|D|}$$

där N_j är antalet dokument i klassen c_j, |*D*|, totala antalet dokument och N_{ij}, är antalet gånger ordet *i*, förekommit i dokumentet från klassen c_j. Approximationen ovan av $P(d_i \mid c_j)$ är dock anpassad till de möjliga värdena 0 eller 1 för d_i, d.v.s. Boolesk viktning gäller i term-dokument-matrisen.

Genom att beräkna $P(c_j \mid d)$, kan alltså sannolikheten för ett visst dokument skall tillhöra respektive klass tas fram (dock omskalad med en okänd faktor eftersom nämnaren tagits bort). För att tilldela en klass väljs då den mest sannolika.

Träningen med naive Bayes, består i att beräkna de sannolikheter som på förhand kan beräknas en gång för alla utifrån träningsdokumenten, d.v.s. alla $P(c_j)$ och $P(d_i \mid c_j)$.

Trots faktumet, att antagandet om villkorligt oberoende generellt sätt inte är sant för ordförekomster i dokument, är Naive Bayes klassificeraren överraskande effektiv.

3.6.3 Vilken maskininlärningsmetod är bäst?

Det bedrivs en ganska omfattande forskning rörande vilken klassificeringsmetod som är bäst. Med några tydliga slutsatser visar sig svåra att göra, då de olika publicerade resultaten inte är direkt jämförbara med varandra. Ofta har olika datakollektioner används, många gånger har man använt olika utvärderingsmått, och ofta har ganska generella kommenterare gjorts utifrån otillräckliga observationer. Även i de fall då den standardiserade kollektionen Reuters använts, går det att använda denna på varierande sätt, och på så sätt få resultat som ej är jämförbara. Men trots detta, finns det en del forskargrupper som gjort standardiserade experiment, till exempel Aas et al[58], Dumais et. al[59], Joachims[60], Yang & Liu[61], och Yang[62]. En huvudslutsats från dessa studier är att Support Vector Machines är en av de bättre metoderna, k-nearest neighbor visar också mycket goda resultat, följt av metoder som naive Bayes, neurala nätverk, Decision Trees, Rocchio, med flera.

[58] Aas & Eikvil: “Text Categorisation: a survey”
[59] Dumais, Platt, Hekerman, & Sahami: “Inductive learning algorithms and representations for text categorization”
[60] Joachims: “Text categorization with Support Vector Machines: Learning with many relevant features”
[61] Yang & Liu: ”A re-examination of text categorisation methods”
[62] Yang: “An evaluation of statistical approaches to text categorization”

3.7 Prestandamätning

En viktig aspekt vid utveckling av klassificerare är hur man mäter dess prestanda. Det finns många olika mått att välja på, där var och en är utformad att mäta någon speciell klassificeringsförmåga hos systemet. I detta stycke beskrivs dem som frekvent rapporteras i litteraturen gällande binär klassificering.

3.7.1 Prestandamätning vid binär klassificering

Ett vanligt tillvägagångssätt vid s.k. flerklass klassificering, är att uppgiften bryts ner i binärt åtskilda klassificerings problem. För varje klass och dokument, avgörs huruvida dokumentet tillhör klassen eller inte. Prestandamätning vid binärklassificering kan sedan beskrivas med hjälp av en två-vägs utfallstabell[63] (tabell 3). Tabellen innehåller fyra celler:

- *a* – antalet dokument *korrekt tilldelade* denna kategori.
- *b* – antalet dokument *inkorrekt tilldelade* denna kategori.
- *c* – antalet dokument *inkorrekt avvisade* denna kategori.
- *d* – antalet dokument *korrekt avvisade* denna kategori.

	JA är korrekt	NEJ är korrekt
Tilldelning JA	a	b
Tilldelning NEJ	c	d

Tabell 3. En utfallstabell som beskriver möjliga utfall vid binär klassificering.

Från dessa kvantiteter definieras sedan följande fem prestandamått:

$$recall = \frac{a}{a+c} \text{ om } a + c > 0, \text{ annars är } recall = 1;$$

$$precision = \frac{a}{a+b} \text{ om } a + b > 0, \text{ annars är } precision = 1;$$

$$fallout = \frac{b}{b+d} \text{ om } b + d > 0, \text{ annars } fallout = 1;$$

$$accuracy = \frac{a+d}{a+b+c+d} \text{ där } a + b + c + d = n > 0;$$

$$error = \frac{b+c}{a+b+c+d} \text{ där } a + b + c + d = n > 0;$$

[63] Ibid.

Break-even point Prestanda måtten presenterade ovan kan dock var något missledande när de betraktas isolerat. Idealet är 1.0 för både recall (täckning) och precision, dock är detta svårt att uppnå och oftast försöker man då finna den bästa kompromissen mellan de två. Om recall och precision trimmas så att de får samma värde, då kallas detta värde för *break-even point* hos systemet[64]. Break-even point är ett vanligt mått vid utvärderingar.

3.7.2 Genomsnittsprestanda vid binär klassificering

Det finns två sätt att mäta genomsnittsprestanda för en binär klassificerare över många klasser, nämligen *macro-average* och *micro-average*[65]. Vid macro-average, används en utfallstabell per klass, och de lokala måtten beräknas först och därefter genomsnittet över klasserna. Vid micro-average, så förenas de individuella klassernas utfallstabeller till en ensam tabell där varje cell för *a*, *b*, *c*, och *d* är summan hos de motsvarande cellerna i de lokala tabellerna. En globala prestanda beräknas sedan från den förenade tabellen. Macro-average viktar prestanda lika hos varje klass, oavsett hur vanlig eller ovanligt förekommande den är. Micro-average, å andra sidan, viktar prestanda lika för varje dokument, och således lyfter den fram prestanda för vanligt förekommande klasser.

Om antalet klasser är *N*, så blir recall och precision vid macro-average:

$$recall = \frac{1}{N}\sum_{i=1}^{N}\frac{a_i}{a_i + c_i}$$

$$precision = \frac{1}{N}\sum_{i=1}^{N}\frac{a_i}{a_i + b_i}$$

och recall och precision vid micro-average:

$$recall = \frac{\sum_{i=1}^{N} a_i}{\sum_{i=1}^{N} a_i + \sum_{i=1}^{N} c_i}$$

$$precision = \frac{\sum_{i=1}^{N} a_i}{\sum_{i=1}^{N} a_i + \sum_{i=1}^{N} b_i}$$

[64] Ibid.
[65] Ibid.

3.8 Testkollektioner

För att objektivt kunna jämföra forskningsresultat inom textklassificering, är det helt avgörande med en standardiserad dokumentkollektion för analys och testning. Två sådana kollektioner som är publikt tillgängliga är:

- REUTERS-21278[66] kollektionen, som består av dokument från nyhetsbyrån Reuters mellan perioden 1987 till 1991;
- OHSUMED [67] kollektionen, som består av titlar och abstrakt från medicinska journaler, vilka i sin tur kommer från MEDLINE databasen.

Dessa två kollektioner, framförallt Reuters-21578, används vid de flesta experiment inom textklassificering. Sebastiani[68], noterar att de flesta experiment som utfördes innan 1997, genomfördes inte under rätt förutsättningar. Sebastiani, lyfter framförallt fram tre punkter som måste tillgodoses för att kunna jämföra olika klassificerare med varandra:

1. samma testkollektion måste användas;
2. samma uppdelning ”splitring” av kollektionen i träning och testning måste göras;
3. samma prestandamått skall användas.

Reuters-21578 används i detta arbete och avsnitt 5 beskriver hur kollektionen har förebehandlats och delats upp för träning och testning.

[66] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[67] Hersh, Buckley, Leone, & Hickman: ”OHSUMED: an interactive retrieval evaluation an new large text collection for research”
[68] Sebastiani, F: “A tutorial on automated text categorisation”

4 AGENTER

4.1 Inledning

Detta kapitel besvarar centrala frågeställningar rörande agenter som: vad är agenter?: vad är intelligens?; hur fungerar agenter?; och varför agenter. Kapitlet avser att översiktligt redogöra för bakomliggande teori som väglett framställningen av ERP Text Miner.

4.2 Vad är en agent?

Det finns många olika definitioner på vad en agent är. Definitioner som varierar från författare till författare. Litteraturen föreslår många olika egenskaper för agenter, och för att summera de olika författarna har en agent minst egenskaperna att vara *autonom*, *reaktiv*, *proaktiv* och *temporärt kontinuerlig*. Men agenter kan även inneha andra egenskaper som förmåga att lära och förmåga samarbeta med varandra. Nedan presenteras först de fyra grundläggande agentegenskaperna. Sedan beskrivs ytterligare några egenskaper som återfinns hos en del agenter men ej hos alla. Materialet är baserat på källorna[69,70,71,72,73,74,75,76].

4.2.1 Grundläggande egenskaper

Autonom
Centralt för en agent är dess förmåga att fungera autonomt, dvs. självständigt. Olikt andra program som mer direkt styrs av användaren, har agenten förmågan att själva ta initiativ och kontrollera sina handlingar och sitt interna tillstånd. Den kan utföra majoriteten av sina uppgifter utan direkt inblandning av dess användare. Autonomi innebär också att agenten ej blint behöver lyda kommandon, utan har förmågan att kunna modifiera förfrågningar, begära förtydliganden, och även vägra vissa kommandon.

Reaktiv
Reaktivitet innebär att en agent har förmågan att kunna upptäcka och reagera på händelser i dess omgivning som är relevanta för dess uppgift. En agent kan dynamiskt välja vilka handlingar som skall utföras samt i vilken sekvens i respons till omgivningens tillstånd.

Proaktiv
Proaktivitet innebär att agenten har ett målorienterat beteende, det vill säga, en förmåga att kunna ta initiativ när det är lämpligt. En agent behöver inte uttryckliga instruktioner från användaren utan kan själv avgöra när och hur den skall uppnå användarens önskemål.

Temporärt kontinuerlig
En agent är en kontinuerlig process. Agenter har sina egna processer eller trådar – det är dessa som möjliggör deras autonomi och proaktivitet. Agenter kan suspenderas för att t.ex. vänta på data eller svar från en annan agent som den sammanbetar med. Om en agent suspenderas eller terminernas, kommer agenten att sparas på sådant sätt att den är möjlig att återskapa eller återstarta.

[69] Bigus & Bigus; "Constructing Intelligent Agents Using Java"
[70] Castelfranchi: "Guarantees for autonomy in cognitive agent architecture"
[71] Etzioni & Weld: "Intelligent agents on the Internet: fact, fiction and forecast"
[72] Franklin & Graesser: "Is it an agent, or just a program? A taxonomy for autonomous agents"
[73] Ingham: "What is an Agent?"
[74] King: "Intelligent Agents: Bring Good Things to Life"
[75] Maes: "Agents that reduce work and information overload"
[76] Wooldrige & Jennings: "Intelligent agents: theory and practice"

Ovanstående fyra egenskaper finns i alla agenter i någon grad. Nedan beskrivs ytterligare några egenskaper som finns i en del agenter men ej i alla.

4.2.2 Ytterligare några egenskaper

Adaptiv
En adaptiv eller lärande agent har förmågan att kunna förändra sitt beteende över tiden. Det kan t.ex. handla om att lära sig nya metoder eller att anpassa sig till omgivningen på något sätt. Typiskt har en adaptiv agent en begränsad kunskapsmassa till att börja med, och förvärvar sedan ny kunskap efter hand.

Kommunikativ
Agenter har ibland förmågan att kunna kommunicera för att kunna uppnå mål eller för att kunna hantera en viss händelse. När agenter skall kunna kommunicera med varandra måste man ta ställning till protokoll samt hur man skall beskriva en domän i termer så att en agent från en annan domän kan förstå.

Kooperativ
Agenter med kooperativ förmåga kan samarbeta med andra agenter. Inte alla agenter som kan kommunicera har kooperativ förmåga – begreppet innebär en förmåga till samarbete mot gemensamt mål, istället för att helt enkelt utbyta information med varandra.

Mobil
En mobil agent utför delegerade uppgifter i ett nätverk. Den kan förflytta sig själv från en maskin till en annan; navigera runt i LAN, WAN eller på Internet för att utföra komplexa uppgifter.

4.3 Vad är intelligens

Med *intelligens* brukar refereras till agenters förmåga att kunna förvärva och tillämpa domänspecifik kunskap för att lösa problem[77]. Sålunda, kan agenter vara relativt dumma, användande av enkel kodad logik, eller relativt sofistikerade, användande av komplex artificiell intelligens (AI) som t.ex. lärande och resonerande.

Intelligenta agenter är till stor del ett resultat från AI-forskningen. Dock är det viktigt att skilja på den mer bredare intelligens som är målet inom AI-forskningen och den intelligens om eftersträvas hos agenter. Det enda egentliga kravet på intelligens som ställs på agenter är att de kan fatta lämpliga beslut inom den tidsram som gäller för att beslutet skall vara användbart. Andra krav på intelligens är beroende på den domän som agenten skall verka inom, t.ex. alla agenter behöver inte kunna lära och resonera.

4.4 Agenttaxonomier

Ett sätt att försöka förstå agentbegreppet är att studera olika indelningsgrunder för agenter. Det finns många olika sätt att klassificera agenter. Ett indelningsgrund är utifrån agentens *processande strategier*. Och en annan är kategorisera agenter utifrån de funktioner som de utför, *processande funktioner*. I följande redogörs kortfattat för dessa två indelningsgrunder.

77 Bigus et al: “Constructing Intelligent Agents Using Java”

4.4.1 Processande strategier

Reaktiva agenter
En av de mer enklare agenttyperna är den reaktiva agenten. Reaktiva agenter saknar interna modeller av omgivningen. De fungerar istället utifrån externa stimuli från omgivningen som upptas från dess sensorer[78,79]. Typiska exempel på denna typ av agenter är robotar.

Proaktiva agenter
Proaktiva agenter innehar domänkunskap och planeringsförmåga som krävs för att kunna utföra en serie handlingar i strävan att nå ett specifikt mål. Proaktiva agenter kan ibland samarbeta med andra agenter för att utföra vissa uppgifter. Oftast används tekniker inom AI som t.ex. resonerande förmåga[80].

Samarbetande agenter
Samarbetande agenter är agenter som samverkar för att lösa problem. Ett viktigt inslag är kommunikationen mellan agenterna, och medan varje individuell agent är autonom, är det synergin från deras samarbete som gör dessa agenter intressanta och användbara. Samarbetande agenter kan lösa problem som ligger bortom förmågan för en enskild agent, och de medger en modularisering genom specialisering av agentfunktioner eller domänkunskap[81].

Mobila agenter
En mobil agent är ett datorprogram som kan förflytta sig mellan olika datorsystem i ett nätverk med syfte att utföra uppgifter för dess användare. En fördel med mobila agenter är att kommunikationen mellan dess hemmasystem och dess fjärrsystem reduceras[82].

4.4.2 Processande funktioner

Kanske den mest naturliga indelningsgrunden för agenter är utifrån de funktioner som de utför. Således, finns det gränssnittsagenter, sökagenter, filteragenter, domänspecifika agenter, etc. I korthet, funktionsspecifika agenter kan skapas genom att kombinera olika agentegenskaper och applicera dessa i olika domäner. Här kommer kortfattat att illustreras två olika generella typer som är mest relevanta för detta arbete: *gränssnitsagenter* och *informationsagenter*.

Gränssnitts agenter
En gränssnittsagent är typiskt en modell av den personliga assistenten. När en agent används med syftet att ge personlig assistens till en användare engagerad i speciell datoruppgift, kallas detta för en gränssnittsagent[83]. En gränssnittsagent bör inneha förmågan att lära, dvs. vara en lärande agent. Den skall gradvis kunna lära sig hur den bättre kan assistera sin användare, genom att observera och imitera sin användare, kunna förstå sin användares intressen, kunna erhålla positiv och negativ feedback, kunna ta emot uttryckliga instruktioner, eller kunna fråga andra agenter om råd. Den grundläggande idén är att genom maskininlärning är det möjligt att skapa gränssnittsagenter som självmant skräddarsyr sitt beteende till användarens behov[84,85].

Gränssnittsagenter kan tjäna sina användare på olika sätt. Till exempel, kan en gränssnittsagent agera som en personlig assistent, som sköter dagliga administrativa sysslor åt sin användare. Den kan

[78] Ibid.
[79] Brooks: "A robust layered control system for a mobile robot"
[80] Bigus et al: "Constructing Intelligent Agents Using Java"
[81] Ibid.
[82] Ibid.
[83] Maes: "Agents that reduce work and information overload"
[84] Ibid.
[85] Mitchell, Caruana, Freitag, McDermott & Zabowski: "Experience with a learning personal assistant"

schemalägga möten, boka resor, hantera email, filtrerar elektroniska nyheter, bevaka eller påminna etc.[86] Gränssnittsagenter kan också används som användarstöd mellan användare och datorsystem. De flesta datorsystem är idag utrustade med väldesignande grafiska gränssnitt, trots detta, kan dessa system vara komplicerade för många människor att använda. En gränssnittsagent kan då kopplas till dessa system och assistera och hjälpa användarna att interagera med systemen. En gränssnittsagent kan också tjäna som koordinator i ett multiagentsystem. Flera specialiserade agenter kan då koordineras genom gränssnittsagenten som då delegerar uppgifter till de specialiserade agenter[87].

Informations agenter
En *informations agent* är en intelligent agent som har access till en eller flera heterogena och geografiskt distribuerade informations källor, och som förvärvar, förmedlar, och underhåller relevant information för dess användare eller andra agenter[88]. Således, förväntas en informations agent tillfredsställa ett eller flera av nedanstående krav[89].

- *Förvärv och förvaltning av information*. Den skall vara kapabel till transparent access till en eller flera olika informationskällor. Den skall kunna samla, extrahera, analysera och filtrerar data, kontrollera källor, och uppdatera relevant information för dess användare eller andra agenter. Generellt innebär förvärv av information en mängd olika scenarier inkluderat avancerad informationsförvärv i databaser samt även inköp av relevant information från elektroniska marknadsplatser.

- *Syntes och presentation av information*. Agenten bör ha förmågan att förena heterogen data och kunna återge mångdimensionella vyer över relevant information för dess användare.

- *Intelligent användarstöd*. Agenten bör dynamiskt kunna anpassa sig till förändringar i användarens önskemål, informationen, samt nätverksmiljön. Den bör kunna ge intelligent, interaktiv stöd till användarens mest vanliga informationsbehov. I detta sammanhang uppträder och fungerar agenten likt en gränssnitsagent.

4.5 Hur fungerar agenter

Hur vet en agent att den skall göra något, eller att den skall reagera på ett visst sätt när något händer? Detta innebär att agenten måste kunna hantera *händelser*, evaluera *tillstånd*, och kunna utföra *handlingar*. Händelser – tillstånd – handlingar definierar det inre maskineriet i en agent, och kallas för *händelse – tillstånds – handlings cykeln*[90].

4.5.1 Händelse-tillstånds-handlings cykeln

En *händelse*, i agentsammanhang, är allting som händer i omgivningen och som agenten bör vara medveten om. En händelse kan till exempel vara att ett nytt email anländer, förändring på webbsida, eller en timer som slår av och på. För att en agent skall kunna upptäcka händelser måste den kunna ta in information från omgivningen. Detta kan göras aktivt genom att skicka meddelanden till andra agenter, eller passivt genom att ta emot en ström av meddelanden från systemet, användaren, eller andra agenter. Information tas in via agentens *sensorer*, vilka kan vara utplacerade i den fysiska världen.

[86] Maes: “Agents that reduce work and information overload”
[87] D´Aloisi & Giannini: “The Info Agent: An Interface for Supporting Users in Intelligent Retrieval”.
[88] Klusch: ”Information agent technology for the Internet: A survey”
[89] Ibid.
[90] Bigus et al: “Constructing Intelligent Agents Using Java”

När en händelse inträffar och fångas upp av agenten, måste den kunna evaluera vad denna händelse innebär och kunna agera riktigt därefter. Detta andra steg, att avgöra tillståndet av omgivningen, kan vara enkelt eller extremt komplext beroende på situationen. Detta funktionssteg kallas för *tillståndsevaluering* i händelse – tillstånds – handlings cykeln.

Efter att agenten identifierat en signifikant händelse och genomfört en tillståndsevaluering, är nästa steg att utföra någon form av *handling*. Denna handling kan vara att upptäcka att ingen handling behöver göras, eller att t.ex. skicka iväg ett meddelande till en annan agent som utför handlingen. Agenter genomför handlingar genom *effektorer*. Intelligenta agenter använder effektorer för sina handlingar, antingen genom att sända meddelanden till andra agenter eller genom att anropa programmeringsgränssnitt eller systemgränssnitt direkt.

En agent bör också ha förmågan att vara proaktiv. Den skall dels kunna reagera på händelser, dels också kunna planera och initiera handlingar själv. Denna egenskap kan ses som en förlängning av händelse – tillstånds – handlings paradigmet. Den planerande handling som agenten utför (meddelandeutskickning, eller gränssnittsanrop) är ett resultat av en tidigare händelse, som försatt agenten i ett planeringsläge.

4.6 Varför agenter?

Det finns många skäl till att använda agenter. Begreppet ”agent” och de olika användningsområdena nämnda i ovanstående textstycken har många paralleller med mänskliga assistenter. Detta assistentliknande beteende går t.ex. att finna i gränssnittsagenter och system där användaren delegerar speciella uppgifter till agenter. Genom att delegera vissa uppgifter till agenter kan användaren avlastas. Användaren sparar tid, tid som kan koncentreras på mer viktigare uppgifter.

Men agenter kan också ha andra fördelar än rollen som personliga assistenter. En tanke är att använda agenter som byggstenar i system. Systemet byggs då upp av en samling agenter som enskild eller gemensamt löser uppgifter autonomt. Ett tillvägagångssätt som framgångsrikt kan tillämpas i många olika system. En del ser detta som ett nytt paradigm inom systemutveckling: Agent-orienterad programmering[91,92].

4.6.1 Agent-orienterad programmering

Agent-orienterad programmering (AOP) är ett ganska naturligt ”nästa steg” från objekt-orienterad programmering (OOP). Inom OOP används modelleringstekniker för att dela upp system i hanterbara delar. Varje del har ett väldefinierat gränssnitt gentemot det övriga systemet, och delarna kan (åtminstone i teorin) konstrueras separat och senare sammanfogas, eller återanvändas när ett liknande behov uppstår.

AOP kan vara ett annat sätt utföra denna systemuppdelning. Delarna är då vanligtvis större (och de i sin tur kan vara uppbyggda enligt OOP), och varje del är en agent som är autonom, reaktiv, proaktiv och temporärt kontinuerlig.

Detta sätt att konstruera är numera mer realistiskt än tidigare. Ramverk med fler-trådade och distribuerade möjligheter är idag tillgängligt för de flesta programmerare på de flesta plattformar, verktyg viktiga för att bygga agenter både snabbt och enkelt.

Design av distribuerade system kan vara mycket komplext beroende på svårigheten att utveckla och implementera protokoll. Med agentabstraktionen kan man kapsla in protokollen i agenter, eller att helt

[91] Parunak: ”Practical and Industrial Applications of Agent-Based Systems”
[92] Jennings and Wooldridge: “Applications of Intelligent Agents”

enkelt låta agenten själv förflytta sig mellan systemen. Med denna mobilitet kan man då slippa omständlig kommunikation.

Att dela upp ett system i agenter kan också ha andra intressanta fördelar. Användare kan till exempel själva ”plugga in” egna agenter i systemet. Detta skulle kunna skapa en marknad av konkurrerande agenter, ”gratis agenter”, istället för att tvinga användare att förlita sig på en specifik systemleverantör för att få ut komplett funktionalitet.

5 EXPERIMENT

5.1 Inledning

Detta kapitel beskriver experiment som genomförts med kollektionen Reuters-21578. Avsikten är att redogöra för hur automatiskt textklassificerar kan utvecklas, samt vilken prestanda som går att uppnå med en sådan.

5.2 Testkollektionen Reuters-21578

För att kunna utveckla en klassificerare, utvärdera den, samt jämföra den med andra klassificerare rapporterade i litteraturen, användes här testkollektionen Reuters-21578 version 1.0, distribution 1.0.

Reuters-21578 är tillgänglig vid adressen:

http://www.daviddlewis.com/resources/testcollections/reuters21578/

Reuters-21578 består av 21,578 artiklar publicerade vid Reuters nyhetsbyrå mellan åren 1987 till 1991. Anställda vid Reuters samlade in artiklarna och indexerade dem under 135 olika klasser. För att möjliggöra att publicerade resultat skulle kunna jämföras mellan olika studier, genomförde David D. Lewis och Stephen Harding en uppmärkning av kollektionen med SGML. För mer detaljer rörande filformat och uppmärkning se Appendix 1.

5.2.1 Förbehandling av kollektionen

Första steget var att förebehandla Reuters så att det var möjligt att dela upp kollektionen i olika standardiserade partitioner. I med att kollektionen är uppmarkerad med SGML, och SGML är mycket svår att bygga mjukvara till, konverterades samlingen till XML.

XML är en förenklad delmängd av SGML – det vill säga en förenklad specifikation på hur man definierar dokumenttyper. Konverteringen till XML innebar ingen dataförlust, utan innebar istället fördelen att det var möjligt att bygga en XML-tolk till kollektionen.

Med denna XML-tolk var det sedan möjligt att extrahera ut dokument och delar ur dokument som kunde användas till träning och testning.

5.2.2 Partitionering av kollektionen i ModApte split

I andra experiment, brukar kollektionen partitioneras i en del för träning och en del för testning. Här följdes tillvägagångssätt som först gjordes av Apte et al[93], den så kallade *ModApte* splittringen, och som även brukar göras av andra forskare, till exempel Aas et al[94]. ModApte ger en standardiserad uppdelning, och fördelen blir att den möjliggör jämförelser med andra resultat.

Enligt dokumentationen skall ModApte leda till 9,603 träningsdokument, och 3,299 testdokument.

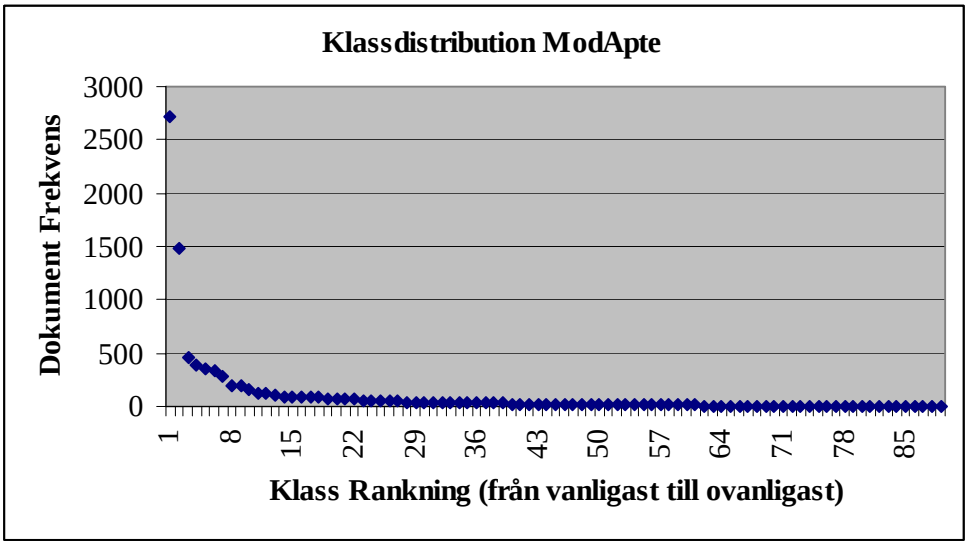
[93] Apte, Damerau, & Weiss: “Automated learning of decision rules for text categorization”
[94] Aas & Eikvil: ”Text Categorisation: a survey”

Men i med att flera dokument antingen saknar klassangivelse (trots att det är märkta med TOPICS=”YES”), eller saknar text inom taggarna <BODY>, </BODY>, togs dessa bort (ett förfarande som även görs av flertalet andra forskare[95]).

ModApte partitionen ledde till följande uppdelning:

- 7,068 träningsdokument, 2,745 testdokument (efter att tagit bort oklassificerade dokument, och dokument som saknar text inom body taggarna).
- 89 stycken testklasser (efter att ha tagit bort klasser som saknar tränings- och testdokument).

Klassdistribution efter denna uppdelning blev då enligt figur 1.



Figur 1. Klassdistribution ModApte split. Antalet träningsdokument per klass. *Källa: Egen*

ModApte har en ganska skev klassfördelning i träningsdelen:

- den mest vanliga klassen ”earn” har en frekvens på 2,709 stycken träningsdokument;
- 79 % av klasserna har mindre än 100 stycken träningsdokument;
- 34 % av klasserna har mindre än 10 stycken träningsdokument.

En komplett översikt för klassdistributionen som använts i detta arbete kan ses i appendix 3 och 4.

5.3 Extrahering av termer

Den text i dokumenten som var inkapslad mellan taggarna <BODY>, </BODY>, konverterades från sitt originalformat (d.v.s. strängar av tecken) till t-dimensionella vektorer som sedan inordnads i ett inverterat index (term-till-dokument-index). I följande understycken beskrives denna procedur närmare.

[95] Ibid.

5.3.1 Förbehandling

Först förebehandlades texten i träningsdokumenten med syfte att ta fram indextermer.

Borttagning av märkord
Med hjälp av den XML-tolk som byggdes, extraherades endast den text som var inkapslad mellan taggarna <BODY>, </BODY>. Således avlägsnades alla märktaggar och all annan metainformation.

Lexikalisk analys
För att kunna identifiera orden i texten togs ett antal beslut rörande siffror, bindesträck, punkteringstecken, samt stora och små bokstäver. Siffror ansågs inte som bra indextermer således togs dessa bort. Ord som var sammanbundna med bindesträck separerades till åtskilda ord. Beträffande punkteringstecken togs dessa bort, sedan omvandlades alla återstående tecken till små bokstäver. Den lexikaliska analysen resulterade i 22,000 unika ord, och 506,652 ord totalt.

Borttagning av stoppord
För att filtrera bort alla stoppord användes en stoppordslista på 887 stycken stoppord (se appendix 2). Detta steg reducerade ner indexstrukturen till 21,374 unika ord och 324,899 ord totalt.

Omvandling till grundform
Alla återstående ord omvandlades sedan till grundform, genom borttagning av suffixdelen. Stemmingsalgoritmen Porters stemmer[96] användes för detta ändamål. Denna process ledde till 15,291 unika ord, och 301,923 ord totalt.

Således, resulterade förbehandlingen i

- 15,291 unika ord (indextermer),
- 301,923 ord totalt, och
- en reducering av indexstrukturen med 6,709 unika ord (från 22,000 till 15,291), d.v.s. 30%.

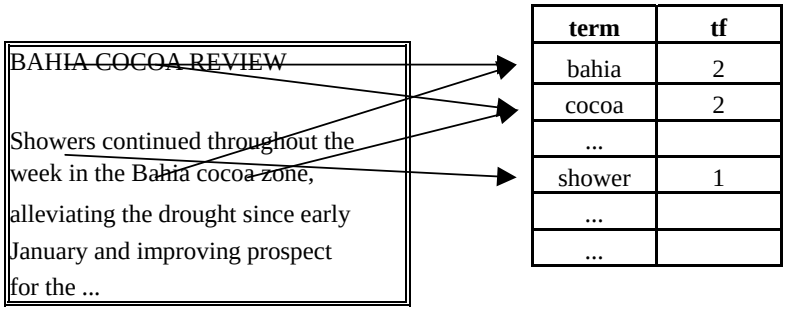
5.3.2 Representation

Andra fasen var att representera de förebehandlade dokumenten. Här användes vektor-rymd-modellen (se avsnitt 3.5.2), där dokumenten uttrycks som t-dimensionella vektorer, och där dessa vektorer sedan organiseras i ett inverterat index (term-till-dokument-index).

Vokabulären och därmed antalet dimensioner hos vektorerna blir här mycket hög, närmare bestämt 15,291 dimensioner innan dimensionsreducering. Men, i med att de flesta dokumenten inte innehåller alla unika ord, blir dessa vektorer glesa. Därför behövs en effektiv metod för att representera glesa vektorer.

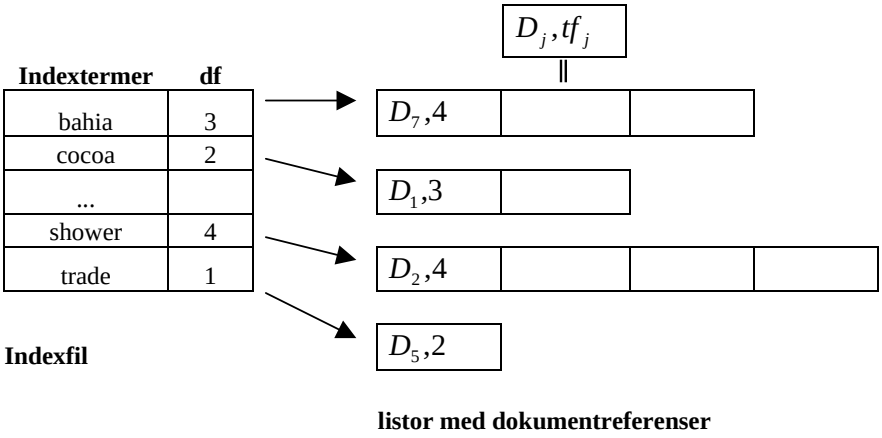
Här implementerades glesa vektorer som hashtabeller (se figur 2). Där nyckeln i tabellen utgörs av termen och värdet utgörs av termens vikt. En fördel med hashtabell som representationsform, är att det tar konstant tid att hitta och uppdatera en vikt för en specifik term. Samtidigt måste man beakta att hela tabellen måste kunna få plats i internminnet.

[96] Porter: ”An Algorithm for Suffix Stripping”



Figur 2. Dokumenten är representerade som term-vektorer (bag-of-words), implementerade som hashtabeller. Nyckeln i tabellen utgörs av ordet (eller termen), och värdet är antalet gånger termen förekommer i dokumentet lagrade som en vikt för termen.

I praktiken lagrades inte term-vektorerna direkt, utan istället användes en inverterad organisation för alla framtagna term-vektorer. En organisationsform som kallas inverterat index, där termerna (nyckelorden) är nycklar och värdena till dessa nycklar utgörs av en lista med dokumentreferenser. Alltså, ett term-till-dokument index, implementerad som en hashtabell (se figur 3).



Figur 3. Ett inverterat index användes för att organisera dokumenten i kollektionen. Termerna i term-vektorerna läggs över i en indexstruktur där via listor med referenser ges de dokument som innehåller termerna.

Algoritmen som användes för att skapa ett inverterat index över dokumenten kan ses i figur 4.

```
Skapa en tom hashtabell, H (d.v.s. ett inverterat index)
För varje dokument, D, (d.v.s. träningsfilerna):
  Skapa en hashtabell, V, för D;
  För varje term, T, i V:
    Om T inte redan är i H, skapa ett TermInfoObjekt för T och lägg in den i H;
    Skapa ett TermFörekomstObjekt för T i D och lägg till den i förekomstlistan
      i TermInfoObjektet för T;
Beräkna IDF för alla termer i H (se avsnittet viktning);
Beräkna vektorlängden för alla dokument i H (se avsnittet viktning);
```

Figur 4. Algoritm för att skapa ett inverterat index.

Med dessa representationsformer, blev tidskomplexiteten för att skapa en term-vektor och indexera ett dokument med n termer O(n). Att indexera m sådana dokument blev då O(m n).

5.3.3 Reducering av dimensioner

Med syfte att spara utrymme, snabba upp samt förbättra träningen, avlägsnades de termer (ord) som ansågs oanvändbara för klassificering. Denna process kallas för feature selection eller reducering av dimensioner och det finns minst fem olika metoder att välja på (se avsnitt 3.5.4).

Här prövades två (separat från varandra) metoder: Document Frequency Thresholding (DFT), samt Mutual Information (MI).

Först prövades DFT, en relativt enkel och samtidigt effektivt metod för att öka klassificeringsprestanda. Dokumentfrekvensen för varje term beräknades och därefter avlägsnades de termer som var under angivet tröskelvärde. Här användes tröskelvärdet 2, vilket resulterade i 8,378 unika termer.

Sedan prövades reducering genom MI. Där MI för en term *w*, med respekt till klassen c_j definierades med följande:

$$MI(w, c_j) \approx \log \frac{A \times N}{(A + C) \times (A + B)}$$

Först beräknades MI poängen för alla termer med respekt till klasserna. Sedan behölls *m* antal termer med högst poäng för respektive klass. Olika värden på *m* testades, det bästa klassificeringsresultatet genererades när *m* var lika med 210 vilket gav 8,147 antal termer.

5.3.4 Viktning

Jag experimenterade med två olika metoder för term-viktning: ltc och tfxidf (se avsnitt 3.5.3 viktning). Således gavs de termer som frekvent förekommer i dokumenten, men övrigt är sällsynta i kollektionen en hög vikt. Vid ltc-viktning togs också hänsyn till att dokumenten kan ha olika längd.

Vid tfxidf-viktning tilldelas vikten, w_{ik} , för termen, *i*, i dokumentet, *k*, enligt följande formula:

$$w_{ik} = tf_{ik} idf_i = tf_{ik} \log\left(N / df_i\right)$$

Där, tf_{ik} , är termfrekvensen av termen *i*, i dokumentet *k*, och idf_i , är den inverterade dokumentfrekvensen av termen *i*.

Och vid ltc-viktning tilldelas vikten w_{ik} , för termen, *i*, i dokumentet, *k*, enligt följande formula:

$$w_{ik} = \frac{\log(tf_{ik} + 1.0) * \log(N / df_i)}{\sum_{j=1}^{M} \left[\log(tf_{jk} + 1.0) * \log(N / df_j)\right]^2}$$

För att beräkna IDF krävdes ytterligare en genomgång över termerna, efter att alla dokumenten hade blivit indexerade. Algoritmen i figur 5 användes för detta.

```
Låt N, vara antalet Dokument totalt;
För varje term, T, i H:
  Bestäm det totala antalet dokument M, där T förekommer (d.v.s. längden på förekomstlistan);
  Sätt IDF för T till log(N/M);
```

Figur 5. Algoritm för att beräkna IDF.

För att sedan med hjälp av cosinuslikhet kunna jämföra olika dokument, behövdes längden hos dokumentvektorerna räknas ut. Längden hos en dokumentvektor är kvadratroten av summan av kvadraterna på vikterna hos dokumentets termer. Då längden räknas ut med hjälp av IDF, blir man tvungen att vänta tills IDF är framräknat, innan vektorlängderna kan beräknas. Algoritmerna i figur 6 och 7 användes vid tfxidf- och ltc-viktade dokumentvektorer.

```
Antag att längden hos dokumentvektorerna är initierade till 0.0;
För varje term, T, i, H:
  Låt, I, vara IDF-vikten hos T;
  För varje TermFörekomst av T i dokumenten D:
    Låt, C, vara termfrekvensen av T i D;
    Räkna upp längden för D med (I * C)^2 ;
För varje dokument D, i, H:
  Sätt längden hos D till kvadratroten av rådande lagrade längd;
```

Figur 6. Algoritm för att beräkna längden hos tfxidf-viktade dokumentvektorer.

```
Antag att längden hos dokumentvektorerna är initierade till 0.0;
För varje term T, i, H:
  Låt, I, vara IDF-vikten för T,
  För varje TermFörekomst av T i dokumenten D:
    Låt, C, vara termfrekvensen av T i D:
    Räkna upp längden för D med (log(C + 1.0) * I)^2;
För varje dokument D, i H:
  Sätt längden för D till kvadratroten av rådande längd;
```

Figur 7. Algoritm för att beräkna längden hos ltc-viktade dokumentvektorer.

Tidskomplexiteten för att beräkna termernas IDF värden för vokabulären *V* (med nämnda representationsformer) blev O(|V|). För att beräkna längden hos vektorerna O(*m n*), d.v.s. samma tidskomplexitet som det tog att indexera *m* dokument.

5.4 Maskininlärningsmetod

k-Nearest Neighbor (kNN) är en metod vilken tidigare har visat mycket god prestanda för textklassificeringsuppgifter[97, 98, 99, 100]. Därför valde jag att använda kNN i detta arbete. En ytterligare fördel blev tillfället att också kunna jämföra mina resultat med andra forskares implementationer.

[97] Aas & Eikvil: "Text Categorisation: a survey"
[98] Joachims: " Text Categorization with Support Vector Machines: Learning with many relevant features"
[99] Weiss, Apté, & Damerau: "Maximizing Text-Mining Performance"
[100] Yang: "An evaluation of statistical approaches to text categorization"

Konstruktion av en klassificerare för klassen $c_i \in C$ består övergripande av två olika faser[101]:

1. definitionen av en klassificeringsfunktion $CSV_i \rightarrow [0,1]$ där givet ett dokument *d*, returernas ett *klassificeringsstatusvärde* för det, d.v.s. ett värde mallen 0 och 1 vilket, grovt uttryckt, representerar beviset huruvida *d* bör klassificeras under c_i.

2. definitionen av en *tröskel* τ_i där $CSV_i(d) \geq \tau_i$, tolkas som beslutet att klassificera *d* under c_i, medan $CSV_i(d) < \tau_i$, tolkas som beslutet att *inte* klassificera *d* under c_i.

kNN byggs med ett s.k. *exempelbaserat* tillvägagångssätt, där dokumentet *d* som skall klassificeras används som en fråga mot träningsdokumenten *Tr*. De klasser där CSV genererare högst klassificeringsstatusvärden anses vara lovande kandidater för att klassificera *d*. Med ett tröskelvärde fattas sedan det slutliga klassificeringsbeslutet (se vidare avsnitt 3.6.1).

Att klassificera ett dokument med kNN innebär att beräkna:

$$CSV_i(d_j) = \sum_{\bar{d}_z \in TR_k(d_j)} RSV(d_j, \bar{d}_z) \cdot ca_{iz}$$

där $TR_k(d_j)$ är en uppsättning av *k* dokument från $\bar{d}_z$, för vilken $RSV(d_j, \bar{d}_z)$ är ett maximum och ca_{iz} är värden från en korrekt beslutsmatris. $RSV(d_j, \bar{d}_z)$ representerar ett likhetsmått mellan dokumenten d_j och $\bar{d}_z$. Här användes cosinuslikhet som mäter vinkeln mellan två vektorer:

$$CosSim(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$

Där w_{ij} är vikten hos termen *i*, i dokumentet *j*, och w_{iq} är vikten hos termen *i*, i det dokument som jämförs.

Vid kNN bestämmer man också ett värde för *k*, vilket indikerar hur många topprankande träningsdokument som skall användas för att beräkna $CSV_i(d_j)$. Värdet för *k* avgörs oftast experimentellt; Yang[102], till exempel, fann att kNN är relativt stabil för ett högt värde på *k*.

För att klassificera en okänd termvektor **d**, jämfördes den först mot alla termvektorerna bland träningsdokumenten. De *k* närmsta träningsvektorerna fanns genom att beräkna cosinus mellan två vektorer. Därefter beräknades ett klassificeringsstatusvärde r_c för klassen *c* utifrån klasserna hos de *k* mest lika träningsvektorerna:

[101] Sebastiani: ” A tutorial on automated text categorisation”
[102] Yang: “An evaluation of statistical approaches to text categorization”

$$r_c(d) = \frac{{}_c\cos(d_i, d)}{\sum_{i=1}^{K} \cos(d_i, d)}$$

Här är $\cos(d_i, d)$ cosinuslikheten mellan grannen *i* och inputdokumentet. Om r_c var högre eller lika med ett givet tröskelvärde så klassificerades inputdokumentet under *c* annars gjordes en avvisning från klassen. När tröskelvärdet ökades minskade klassificerarens täckning (recall) och precisionen ökade (Se avsnitt 5.5).

5.5 Utvärderingsresultat

I detta avsnitt presenterar jag erhållna utvärderingsresultat. Då jag experimenterade med två olika vikningsprinciper (tfxidf och ltc), och då jag önskade jämföra mina resultat med state-of-the-art, presenteras resultaten i tre underavsnitt.

För att hitta ett lämpligt värde för *k*, d.v.s., för att hitta hur många topprankade träningsdokument som skulle användas för att beräkna klassificeringsstatusvärde, experimenterade jag med olika värde för *k*. Jag prövade *k* < 30, 30, 45, 65. Jag fann att prestanda för min kNN var relativ stabil för stora värden för *k*, alltså samma slutsats som Yang[103]. De resultat som återges här genererades när *k* = 45.

För att reducera dimensioner prövade jag två olika metoder: *Document Frequency Thresholding* (DFT), samt *Mutual Information* (MI) (se avsnitt 3.5.1). Jag fann att både metoder genererade ganska likvärdiga utvärderingsresultat. Här presenter jag de resultat som erhölls med DFT, då jag anser att DFT var bättre än MI av flera skäl. För det första, genererade DFT och MI ungefär likvärdiga utvärderingsresultat. För det andra, var DFT betydligt enklare att implementera och att använda. För det tredje, var tids- och rymdkomplexiteten för DFT lägre vilket gav betydligt snabbare exekvering.

Vid DFT experimenterades dels med blygsam dels med aggressiv dimensionsreducering. Jag fann att klassificerarens recall var stabil oavsett antalet reducerade dimensioner, men att klassificerarens precision och därmed systemets break-even point, förbättrades ju mer jag reducerade ner vokabulären. Resultaten som presenteras med ltc-viktning erhölls efter att jag tagit bort alla termer vars dokumentfrekvens var under 11, vilket gav 2,783 unika termer. Resultaten som presenteras med tfxidf-viktning erhölls efter att jag tagit bort alla termer vars dokumentfrekvens var under 5, vilket gav 4,495 unika termer.

För att kunna utvärdera klassificerarnas prestanda för respektive klass, använde jag de fem prestandamåtten: *recall*, *precision*, *fallout*, *accuracy*, och *error* (se avsnitt 3.7.1). För att kunna utvärdera genomsnittprestanda för alla klasser använde jag *micro-averaging* (se avsnitt 3.7.2).

Då jag önskade så höga värden som möjligt för både recall och precision blev jag först tvungen att hitta det tröskelvärde som gav den bästa kompromissen mellan de två. Eftersom recall ökar med minskat tröskelvärde och precision minskar med detsamma, blev jag tvungen att söka efter den punkt där båda är lika höga, d.v.s. systemets *break-even point* (se avsnitt 3.7.1). För att hitta break-even punkten använde jag micro-averaging.

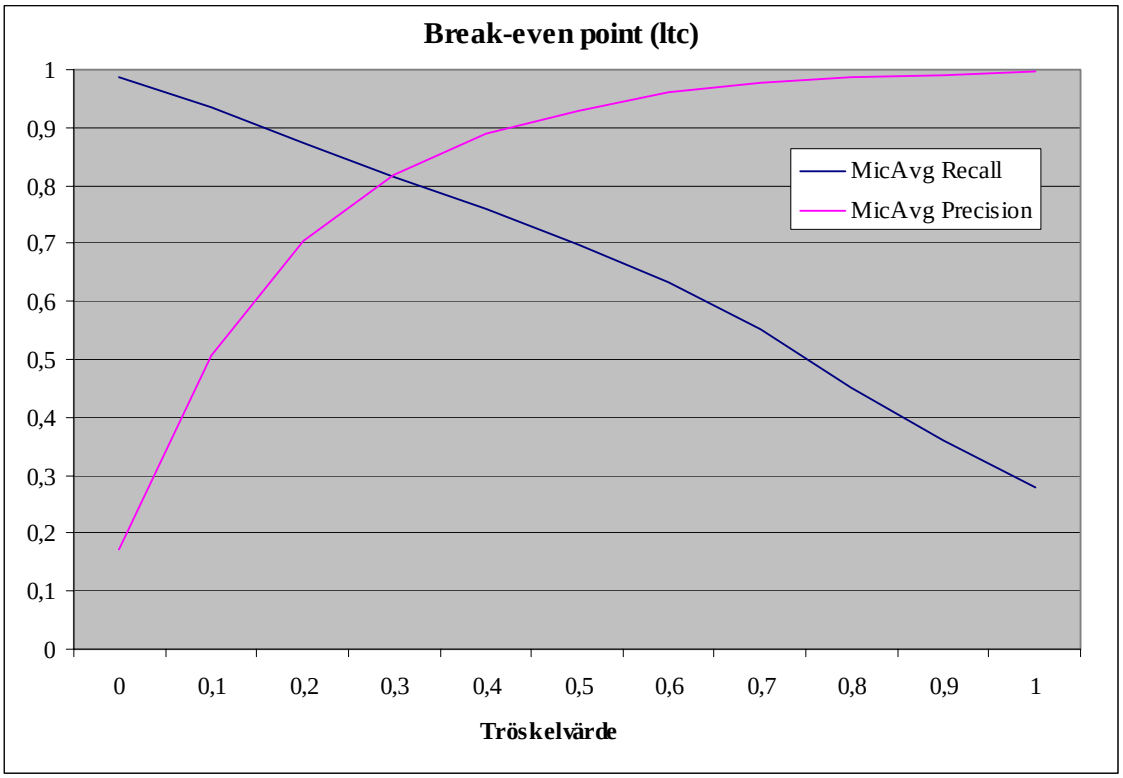
Ovan nämnda prestandamått beräknades dels med ltc-viktning och dels med tfxidf-viktning (se nedanstående underavsnitt). Det bästa resultatet genererades med ltc-viktning. Detta resultat jämfördes med tidigare publicerade resultat (se avsnitt 5.5.3).

[103] Ibid.

5.5.1 Resultat med ltc-viktning

Figur 8 återger micro-averge recall och micro-average precision för olika tröskelvärden med ltc-viktning. Vid tröskelvärdet 0,3 hittades klassificerarens break-even point. Micro-average recall och micro-average precision var då 0,816 och 0,818, vilket indikerar en break-even point på 0,81.

När endast de tio största klasserna testades, vilka är listade i tabell 4, blev recall/precision break-even point 0,90, vilket uppnåddes vid tröskelvärdet 0,4.



Figur 8. Klassificerarns break-even point för alla klasser med ltc-viktning var 0,81 vid tröskeln 0,3.

Tabell 4 återger resultaten för respektive klass med ltc-viktning vid klassificerarens break-even point som uppnåddes vid tröskelvärdet 0,3. Av utrymmesskäl återges här endast de tio största kategorierna. För komplett tabell se appendix 3.

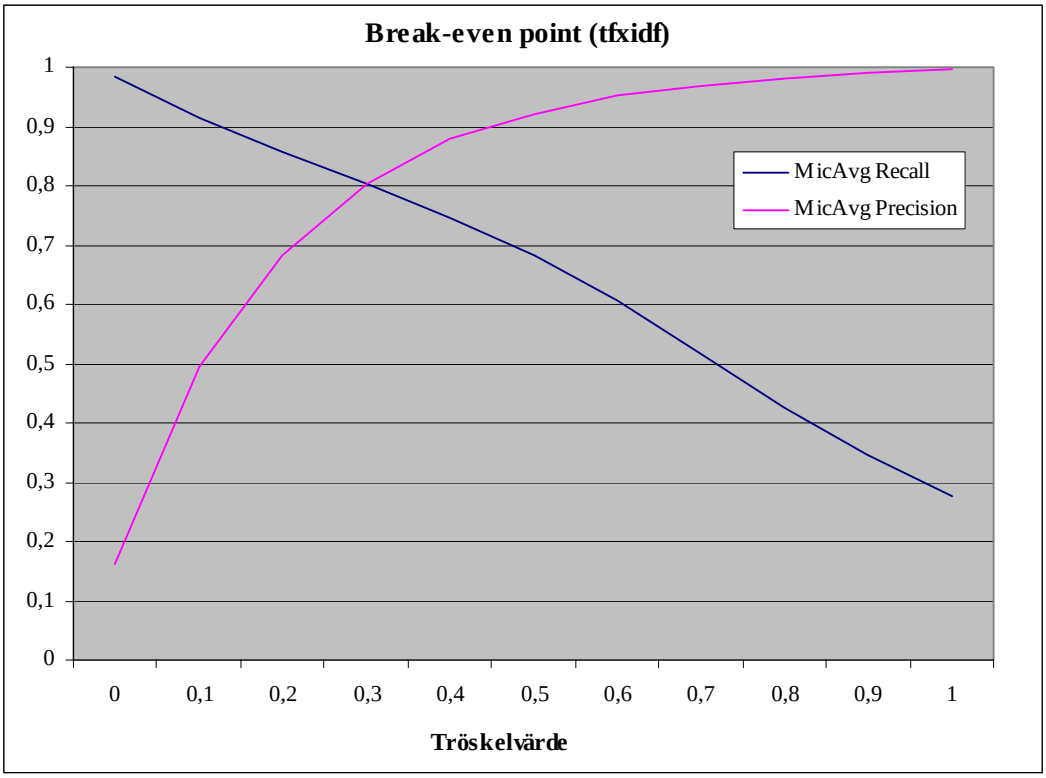
Topic	*Nof train*	*Nof test*	*Recall*	*Precision*	*Fallout*	*Accuray*	*Error*
earn	2709	1045	0,995	0,874	0,088	0,944	0,056
acq	1488	643	0,955	0,936	0,02	0,974	0,026
money-fx	461	142	0,958	0,673	0,025	0,974	0,026
grain	395	134	0,903	0,716	0,018	0,978	0,022
crude	351	161	0,932	0,777	0,017	0,98	0,02
trade	337	112	0,839	0,653	0,019	0,975	0,025
interest	289	102	0,804	0,719	0,012	0,981	0,019
wheat	198	66	0,909	0,6	0,015	0,983	0,017
ship	192	85	0,835	0,816	0,006	0,989	0,011
corn	161	48	0,813	0,629	0,009	0,988	0,012

Tabell 4. Resultat för de 10 största klasserna vid klassificerarens break-even point som uppnåddes vid tröskelvärdet 0,3. Kolumnerna *Nof train* och *Nof test* anger antalet dokument för träning och testning.

5.5.2 Resultat med tfxidf-viktning

Figur 9 återger micro-averge recall och micro-average precision för olika tröskelvärden med tfxidf-viktning. Vid tröskelvärdet 0,3 hittades klassificerarens break-even point. Micro-average recall och micro-average precision var då 0,80 och 0,80, vilket ger en break-even point på 0,80.

När endast de tio största klasserna testades, vilka är listade i tabell 5, blev recall/precision break-even point 0,88, vilket uppnåddes vid tröskelvärdet 0,4.



Figur 9. Klassificerarns break-even point för alla klasser med tfxidf-viktning var 0,802 vid tröskeln 0,3.

Tabell 5 återger resultaten för respektive klass med tfxidf-viktning vid klassificerarens break-even point som uppnåddes vid tröskelvärdet 0,3. Av utrymmesskäl återges här endast de tio största kategorierna. För komplett tabell se appendix 4.

Topic	*Nof train*	*Nof test*	*Recall*	*Precision*	*Fallout*	*Accuray*	*Error*
earn	2709	1045	0,994	0,835	0,121	0,923	0,077
acq	1488	643	0,916	0,944	0,017	0,968	0,032
money-fx	461	142	0,944	0,62	0,032	0,967	0,033
grain	395	134	0,866	0,72	0,017	0,977	0,023
crude	351	161	0,932	0,777	0,017	0,98	0,02
trade	337	112	0,83	0,694	0,016	0,978	0,022
interest	289	102	0,833	0,708	0,013	0,981	0,019
wheat	198	66	0,833	0,579	0,015	0,981	0,019
ship	192	85	0,8	0,81	0,006	0,988	0,012
corn	161	48	0,729	0,686	0,006	0,989	0,011

Tabell 5. Resultat för de 10 största klasserna vid klassificerarens break-even point som uppnåddes vid tröskelvärdet 0,3. Kolumnerna *Nof train* och *Nof test* anger antalet dokument för träning och testning.

5.5.3 Jämförelse med tidigare publicerade resultat

I detta avsnitt jämför jag min klassificerare med andra klassificerare rapporterade i litteraturen. För att återigen referera till Sebastiani[104], så måste framförallt tre kriterier uppfyllas för att kunna jämföra klassificerare med varandra:

1. samma dokumentkollektion måste användas;
2. samma uppdelning ”splitring” av kollektionen i träning och testning måste göras;
3. samma prestandamått skall användas.

De resultat som jämförs här uppfyller dessa kriterier samt representerar ”state-of-the-art” inom textklassificering. Resultaten är framtagna av Aas et al[105], samt Yang[106], tabell 6 och 7 summerar deras samt detta arbete. Alla arbeten har använt kollektionen Reuters-21578 med ModApte uppdelningen. Den första tabellen återger detta, samt anger hur författarna har viktat termer, reducerat dimensioner, typ av klassificeringsuppgift, samt vilket mått som använts för utvärdering.

Aas et al., använde s.k. entropy-viktning, en metod baserad på informationsteori och som anses vara den mest sofistikerade viktningsmetoden (närmare beskrivning av metoden ges i Aas at el). Detta arbete samt Yang har använt ltc-viktning (se avsnitt 3.5.3).

För att reducera dimensioner använde Aas et al Document Frequency Thresholding (DFT) och reparameterisering med Latent Semantic Indexing (LSI). Först avlägsnades de ord som endast förekom i ett dokument, vilket gav dem 8,315 unika ord. Därefter genomfördes LSI på den resterande 8,315 x 7,063 term-dokument-matrisen (se vidare Aas et al.). I detta arbete användes DFT. Alla termer vars dokumentfrekvens var under 11 togs bort vilket gav 2,783 unika termer. Vokabulären reducerades med 82 %. Yang använde X^2-statistik (se avsnitt 3.5.4) och fick 1 – 2 % bättre prestanda beträffande precision och break-even, efter att reducerat vokabulären med 85 %.

Alla arbeten fokuserat på binära klassificeringsuppgifter, där ett dokument klassificeras som antingen relevant eller orelevant med hänsyn till fördefinierade klasserna. Vad gäller prestandamått har alla författare använt micro-average recall/precision break-even point.

[104] Sebastiani: “A tutorial on automated text categorisation”
[105] Aas & Eikvil: “Text Categorisation: a survey”
[106] Yang: “An evaluation of statistical approaches to text categorization”

Tabell 7 listar författarnas maskininlärningsmetoder, samt den uppnådda break-even punkten. Alla författare har testat kNN. Yang rapporterar resultat för fler metoder listade här, men jag har valt göra ett urval från den bästa klassificeraren (vilket var kNN) till den sämsta (vilket var Word).

Författare	*Kollektion*	*Uppdelning*	*Viktning*	*Reducering*	*Uppgift*	*Mått*
Aas & Eikvil	Reuters-21578	ModApte	entropy	DFT & LSI	Binär	Break-even
Plenk	Reuters-21578	ModApte	ltc	DFT	Binär	Break-even
Yang	Reuters-21578	ModApte	ltc	X^2	Binär	Break-even

Tabell 6. Summering av detta arbete samt tidigare arbeten, där standardiserad testning utförts med dokumentkollektionen Reuters-21578. Tabellen återger kollektionens uppdelning, term-viktnings principer, metoder för att reducera dimensioner, klassificeringsuppgift, samt prestandamått för utvärdering. Den mellersta raden återger detta arbete.

Författare	*kNN*	*Ripper*	*DTree*	*Experts*	*Rocchio*	*Bayes*	*Word*
Aas & Eikvil	.80	-	-	-	-	-	-
Plenk	.81	-	-	-	-	-	-
Yang	.85	.80	.79	.76	.75	.71	.29

Tabell 7. Summering av prestandaresultat uttryckt med micro-average recall/precision break-even points. Sträcket ”-” anger att metoden inte testades av författaren. Den mellersta raden återger resultatet för detta arbete.

Utifrån de resultat som återges här, kan ett några intressanta slutsatser göras (se Yang för en mer detaljerad diskussion rörande hennes arbete):

- kNN tycks vara en av de bästa klassificeringsmetoderna.
- Resultatet för min kNN är helt i klass med ”state-of-the-art”.
- Jag fick något bättre resultat än Aas et al, trots att de använde mer sofistikerade metoder för att vikta termer och för att reducera dimensioner. Detta antyder att ltc-viktning (den viktningsprincip jag använde) verkar fungerar minst lika bra.
- Yang rapporterar något högre break-even för sin kNN än för min. Yang har använt X^2-statistik för dimensionsreducering. Detta antyder att val av dimensionsreducerande teknik tycks vara kritiskt för att ytterligare öka klassificeringsprestanda.
- Naive Bays och Rocchio, två klassificeringsmetoder som frekvent rapporteras i litteraturen, tycks vara de sämsta metoderna.
- Det lägsta prestandaresultatet är rapporterat för Word, en klassificerare som implementerades av Yang och som saknar lärande förmåga. Detta indikerar att maskininlärning är den teknik som bör användas för att bygga automatiska textklassificeringssystem.

För övrigt kan diskuteras om en break-even punkt på 0.81 är tillräckligt bra för att kunna accepteras av en slutanvändare. Här kan då påpekas att en utvärdering med Reuters-21578 i ModApte uppdelning, bör vara bland de svåraste testerna för en klassificerare. ModApte har en skev fördelning av tränings- och testdokument per klass, vid varje klassificeringsbeslut skall nära 100 olika klasser övervägas (se avsnitt 5.2.2). När de anställda vid Reuters en gång kategoriserade dokumenten, verkar det också som en del klassificeringar gjordes mindre noga. Slutligen kan nämnas att jag även testade min

klassificerare på egna sammansatta dokumentkollektioner. De resultat som då erhölls var nära 1,0 för både recall och precision, alltså resultat som den mest kräsne slutanvändare kan önska sig.

6 ETT AGENTSYSTEM FÖR AUTOMATISK TEXTKLASSIFICERING

6.1 Inledning

Detta kapitel beskriver utvecklingen av ERP Text Miner. Avsikten är att redogöra för de designbeslut som fattades under arbetet samt att demonstrera hur systemet kan användas.

6.2 ERP Text Miner

6.2.1 Systembeskrivning

ERP Text Miner (ETM) är ett agent baserat system, helt utvecklat i Java, med förmågan att förvärva och klassificera dokument. ETM stödjer hela textklassificeringsprocessen, från framtagning och utvärdering av textklassificeringsmoduler till tillämpning av dessa moduler i ett agentbaserat system.

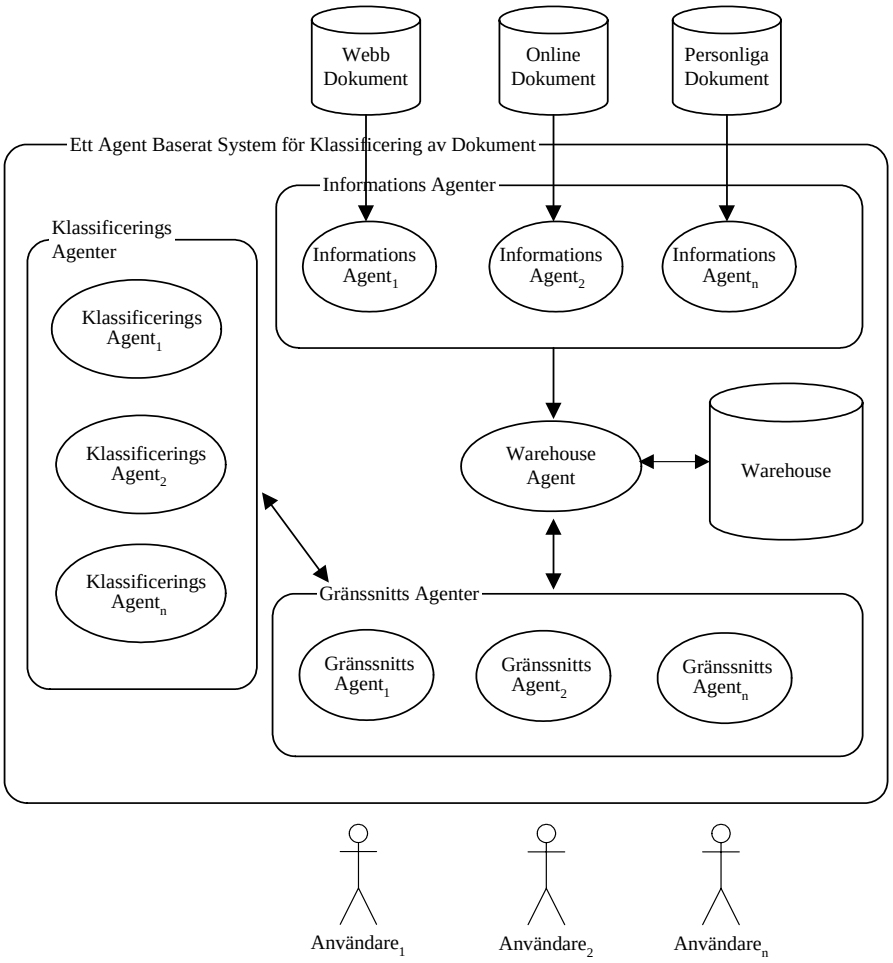
Dagligen produceras stora mängder textbaserad information, dels av enskilda individer dels av organisationer. Det kan vara: email, PM, rapporter, nyhetsartiklar, kundundersökningar, patent information, medicinska journaler etc. Problemet är att det inte finns tid att läsa allt, än mindre att utforska och klassificera dokumenten så att värdefull information kan tillvaratas. ETM syftar till att tillfredsställa detta behov.

Alla som behöver söka igenom stora volymer av text kan ha nytta ETM. Behovet kan till exempel vara att hitta relevant information för ett specifikt beslut, eller för att identifiera problem, möjligheter och trender. Potentiella användare inkluderare chefer, försäljare, marknadsförare, tekniker, support personal, personaladministratörer, analytiker, forskare, läkare, jurister m.fl. Exempelvis kan företag som hanterar stor del av sin kundkontakt via email använda ETM att prioritera mailen för snabbare uppföljning. Personaladministratörer kan använda ETM att strukturera tusentals personprofiler för att matcha kvalificerade sökande med lediga tjänster. Läkemedelsföretag kan komplettera kliniska tester med analys av medicinska artiklar som kan laddas ner från dokumentdatabaser som till exempel MEDLINE.

ETM erbjuder dels ett träningsverktyg där användare kan ta fram och uttesta textklassificeringsmoduler för valfria dokumentkollektioner, dels ett agentbaserat system som koordinerar en serie agenter för förvärv och klassificering av dokument.

6.2.2 Arkitektur

Systemet består av fyra typer av agenter: *informationsagenter* som förvärvar textuell information från interna och externa informationskällor samt bevakar källor för ändringar; *warehouseagent* som förvaltar förvärvad information i en dokumentdatabas; *klassificeringsagenter* som klassificerar insamlade dokument; *gränssnittsagenter* som sköter kommunikationen mellan användare och de olika agenterna i systemet. Arkitekturen för systemet återges i figur 9.



Figur 9. Arkitekturen för systemet.

Systemet är ett mulitagentsystem vilket innebär att det är möjligt att skapa flera instanser av varje agentttyp. Varje agenttyp har en unik huvuduppgift vilket tydligt urskiljer agenterna från varandra. Namnet för agenttypen återger dess syte vilket här indikerar en hög grad av samhörighet i systemet. Agenterna kommunicerar med varandra via agentmeddelanden genom ett enhetligt systemgränssnitt. Detta innebär här att systemet har en låg grad av koppling.

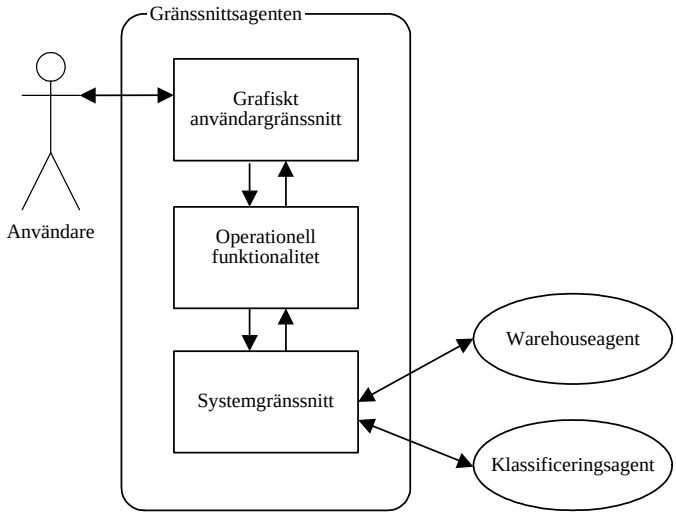
Alla agenter har följande fem egenskaper gemensamt:

Autonoma	Agenterna kan operera utan direkt inblandning av användaren.
Reaktiva	Agenterna kan känna av sin omgivning och reagera på lämpligt sätt mot de förändringar som inträffar.
Proaktiva	Agenterna verkar inte endast mot förändringar i omgivningen, utan de kan också uppträda målorienterat genom att ta initiativ.
Temporärt kontinuerliga	Agenterna har egna trådar som möjliggör deras autonomi och proaktivitet.
Kooperativa	Agenterna kan samverka med andra agenter i systemet för att uppnå särskilda mål.

Nedan beskrivs de olika agenterna lite mer ingående.

6.2.3 Gränssnittsagenterna

Interaktionen mellan användare, klassificeringsagenter och warehouseagenten görs genom gränssnittsagenter. En gränssnittsagent fungerar alltså som en medlare mellan dessa entiteter. Arkitekturen hos en gränssnittsagent består av tre moduler vilket återges i figur 10.



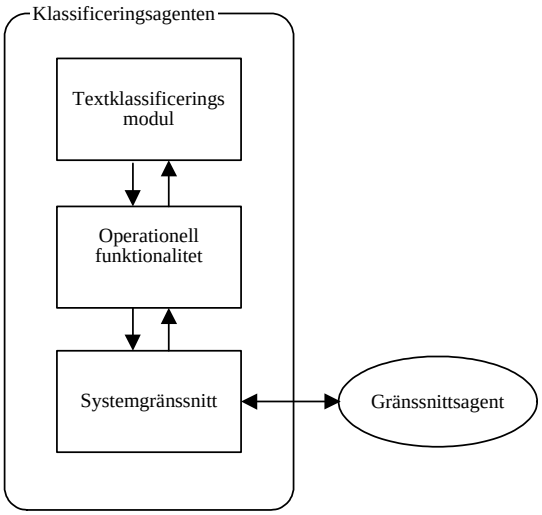
Figur 10. Arkitekturen för gränssnittsagenten.

Via det grafiska gränssnittet kan användaren ge kommandon till gränssnittsagenten. Kommandon kan till exempel vara begäran om hämtning av dokument eller begäran om klassificering av dokument. Den operationella funktionaliteten översätter kommandot till ett agentmeddelande som sedan via systemgränssnittet skickas över till berörd agent. Om önskemålet till exempel är klassificering kommer klassificeringsagenten att skrida till verket. Resultat skickas sedan tillbaka till gränssnittsagenten som via det grafiska gränssnittet presenterar dokumenten för användaren.

En användare kan registrera flera gränssnittsagenter och då följaktligen flera klassificeringsagenter. På detta sätt kan användaren parallellt strukturera flera olika dokumentdomäner, var och en med ett skräddarsytt gränssnitt.

6.2.4 Klassificeringsagenterna

Klassificeringsagenten har som namnet antyder uppgiften att klassificera insamlade dokument under fördefinierade kategorier. Arkitekturen för klassificeringsagenten återges i figur 11.

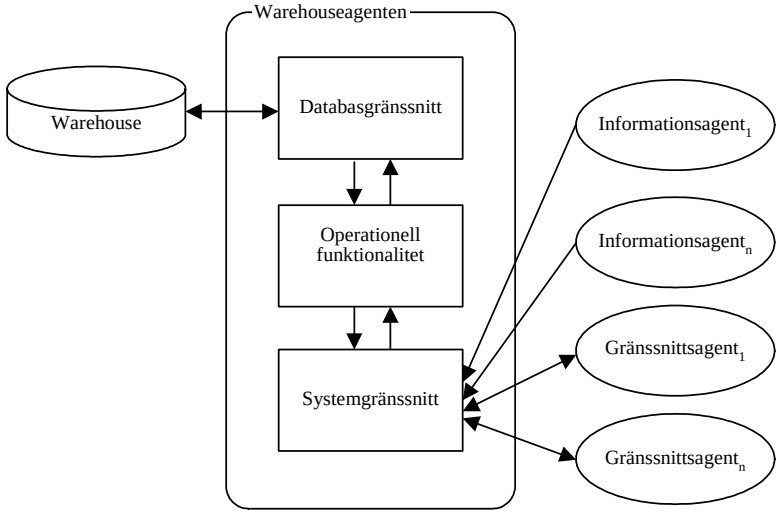


Figur 11. Arkitekturen för klassificeringsagenten.

Klassificeringsagenten består av tre moduler. Systemgränssnittet sköter kommunikation mellan gränssnittsagenten. När klassificeringsagenten får ett meddelande från gränssnittsagenten klassificeras aktuellt dokument med laddad textklassificeringsmodul. När klassificeringen är klar märks dokumentet med den eller de klasser som det skall kategoriseras under. Ett agentmeddelande sätts ihop av den operationella modulen som sedan skickar detta via systemgränssnittet till gränssnittsagenten som då uppdaterar användargränssnittet.

6.2.5 Warehouseagenten

Warehouseagenten förvaltar insamlade dokument i en dokumentdatabas. Warehouseagenten syftar till att ge transparent access till en centraliserade dokumentkälla. Warehouseagenten fungerar som en wrapper som informationsagenter och gränssnittsagenter kan kommunicera med. Arkitekturen för warehouseagenten består av tre moduler vilket återges i figur 12.



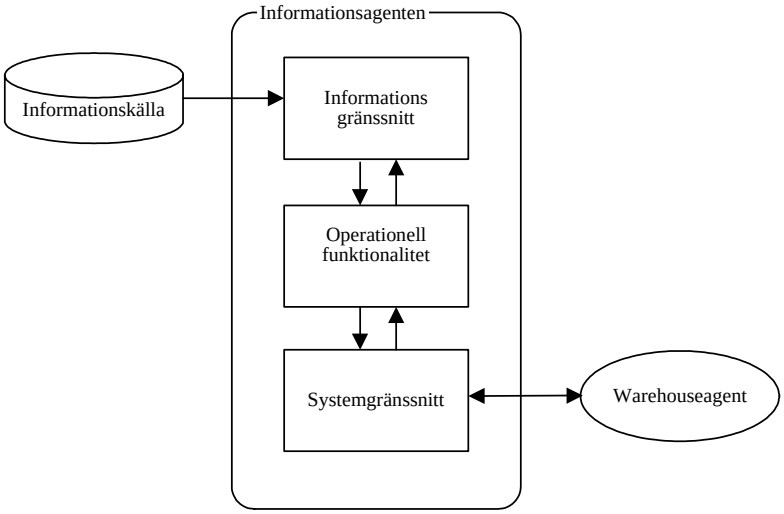
Figur 12. Arkitekturen för warehouseagenten.

Systemgränssnittet sköter kommunikationen mellan en till flera registrerade informationsagenter och gränssnittsagenter. Via systemgränssnittet fås olika agentmeddelanden som till exempel kan vara begäran om dokument eller inläggning av dokument i databasen. Den operationella modulen

översätter agentmeddelandena till databasanrop och passar över dessa till databasgränssnittet. Databasgränssnittet översätter sedan anropen till JDBC anrop som passas över via en JDBC-driver till dokumentdatabasen.

6.2.6 Informationsagenterna

Informationsagenterna har som uppgift att förvärva dokument från interna och externa informationskällor samt att bevaka dessa källor för ändringar. Arkitekturen för informationsagenten består av tre moduler vilket återges i figur 13.



Figur 13. Arkitekturen för informationsagenten.

Via informationsgränssnittet fås access till olika informationskällor vilka kan vara heterogena och distribuerade över nätverk. Dokument som förvärvas transformeras via den operationella modulen till ett standardiserat format. Systemgränssnittet skickar sedan över detta standardiserade dokument till warehouseagenten som då lägger in det i databasen.

6.3 Demonstration av systemet

I detta avsnitt demonstreras ERP Text Miner vars design diskuterades ovan. Som tidigare nämnts, kan alla som snabbt behöver hitta relevant information i stora textvolymer ha nytta av ETM. Här väljer jag att demonstrera ETM med ett användarescenario inom värdepappershandel.

6.3.1 Användarscenario

Låt oss säga att vi har en investerare som dagligen gör affärer om köp och försäljning av aktier, eller liknande värdepapper.

Framgång i affärer bygger på kunskap. Ju bättre informerad om företag, konjunkturer och marknadens sätt att reagera, desto bättre är chanserna att göra lyckade affärer. Värdefull information förekommer i många olika sammanhang och i många olika former. Ofta måste det ”läggas pussel” med olika informationsbitar för att få en så fullständig bild av konjunkturförlopp, företag och värdepapper som möjligt. Informationen som måste sökas i olika källor.

Alltså det finns en positiv korrelation mellan information rörande marknadshändelser och hur attraktivt ett värdepapper är som investeringsobjekt. Men, på grund av den stora volymen av sådan information, är det omöjligt att hitta och att läsa igenom allt. Därför skulle det vara mycket värdefullt med system som automatiskt kan klassificera information i kategorier som indikerar dess innebörd.

Investeraren har just märkt att det händer intressanta saker på råvarubörserna runt om i värden. Guldpriset noterade sjuårshögsta på råvarubörsen i London. Utbudet på olja har sjunkit vilket fått oljepriset att fluktuera kraftigt. Priset på vete har också gått upp på spannmålsbörsen i Chicago. Investeraren undrar därför om det kan vara lönsamt att flytta över sina positioner i råvaror i stället för aktier. Han beslutar sig därmed att ta fram en textklassificerare som kan hjälpa honom att undersöka saken närmare.

6.3.2 Exempel för tränig och testning

Framtagning av en klassificerare går ut på att klassificeraren lär sig karaktäristiken hos en uppsättning dokument som tidigare har klassificerats av en expert. Investeraren börjar således först med att sätta ihop en sådan kollektion. Han beslutar sig att dokument inom följande klasser är relevanta just för stunden:

Klasser	**Dokument som handlar om**
Crude	Olja
Gas	Gas
Livestock	Kreatur
Carcass	Slaktkroppar
Corn	Majs
Grain	Spannmål
Wheat	Vete
Copper	Koppar
Gold	Guld
Interest	Räntor
Money	Växlingskurser
Acquisitions	Företagsförvärv
Earnings	Vinstredovisning
Trade	Handel
Shipments	Frakter

Han väljer ut ett antal dokument för respektive klass och placerar dem i en katalog för träning och testning. I katalogen lägger han också en datasetfil där han skriver namnen på de klasser som gäller.

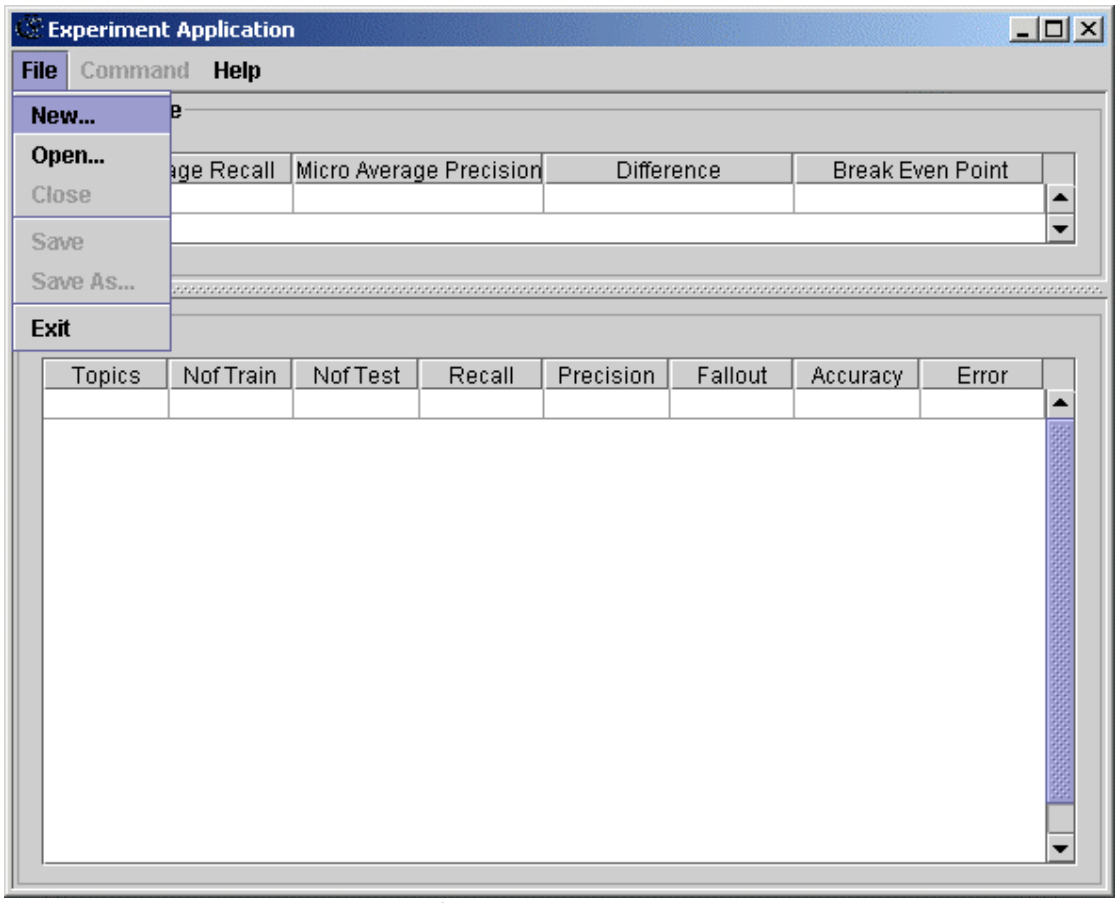
6.3.3 Framtagning av klassificeringsmodul

Efter att sammanställt träningskollektionen är det dags att starta träningsverktyget som ingår i ERP Text Miner. Träningsverktyget är en applikation där användaren kan ta fram och uttesta klassificeringsmoduler för valfria dokumentkollektioner. Verktyget stödjer hela framtagningsprocessen och har följande egenskaper.

- Grafiskt användargränssnitt.
- Access till olika textformat som ASCII, HTML och XML.
- Stöd för engelska.
- Förbehandling av text med lexikalisk analys, stoppordsborttagning och stemming.
- Dimensionsreducering med Document Frequency Thresholding.

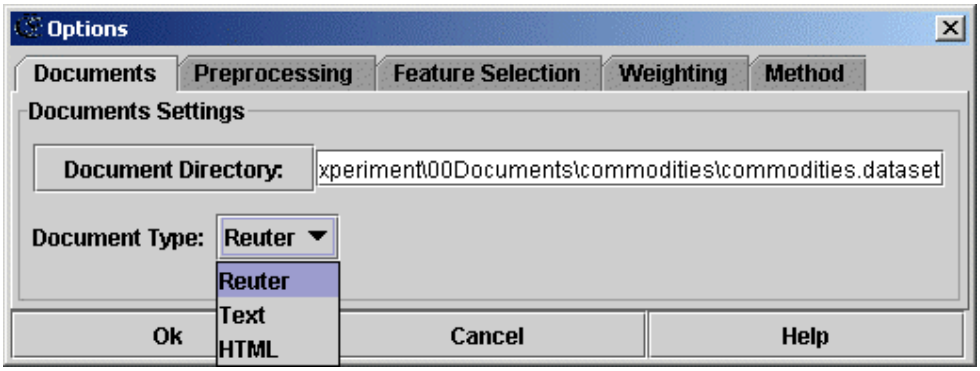
- Termviktning med TFxIDF eller LTC.
- Klassificeringsmetoden k-Nearest Neighbors.
- Prestandamätning dels för enskilda klasser dels med micro-average recall/precision och break-even point.

Investeraren startar upp verktyget och går in under filmenyn och väljer nytt experiment (se figur 14).

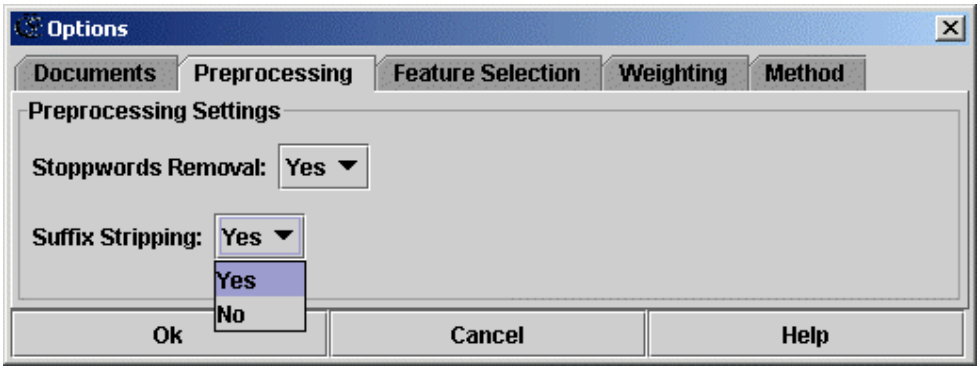


Figur 14. Träningsverktyget som ingår i ERP Text Miner. Första steget är att skapa ett nytt experiment genom att välja ”New…” under filmenyn.

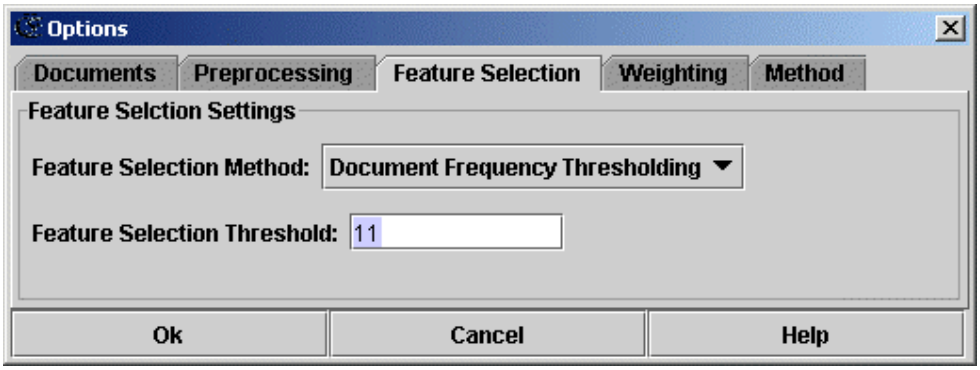
Nästa steg är att kalibrera verktyget för träning och testning. Detta görs genom att gå in under kommandomenyn och välja ”Options”. En dialog med fem stycken flikar poppar up. Under flikarna anges aktuell dokumentkollektion och dess typ, hur kollektionen skall förbehandlas, inställning av feature selection, termvikningsprincip och klassificeringsmetod. Figurerna 15 till 19 visar detta förlopp.



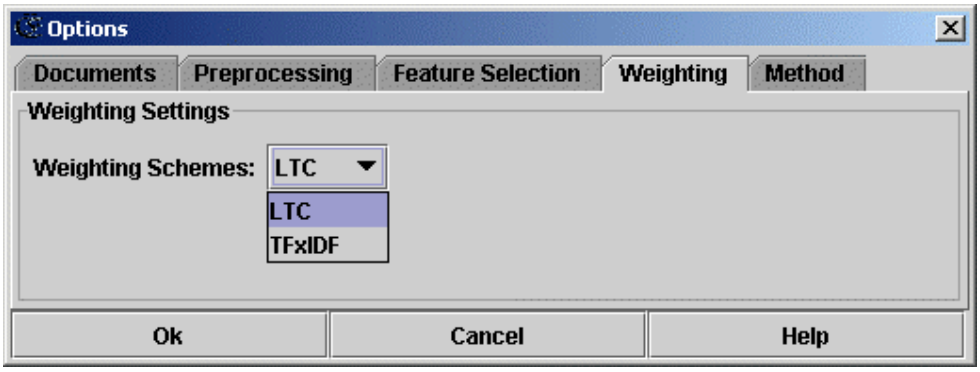
Figur 15. Under dokumentfliken väljs den dokumentkollektion som skall användas för träning och testning samt dess typ.



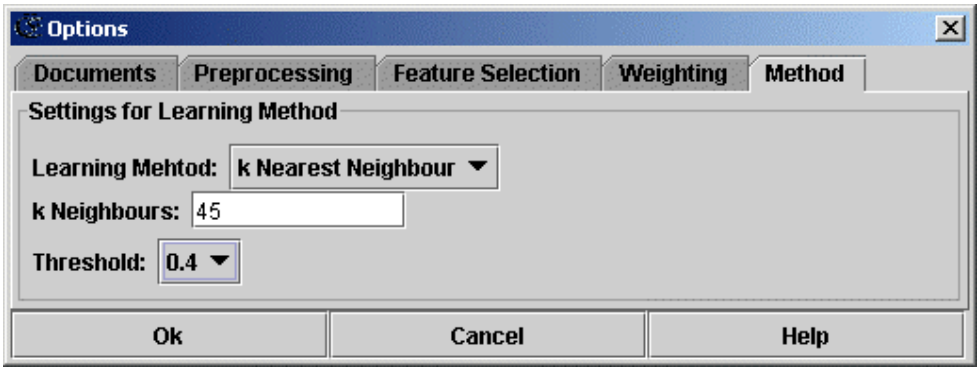
Figur 16. Under förbehandlingsfliken väljs hur dokumenten skall förbehandlas gällande stoppordsborttagning och omvandling till grundform.



Figur 17. Under fliken feature selection väljs metod att reducera dimensioner samt tröskelvärde.



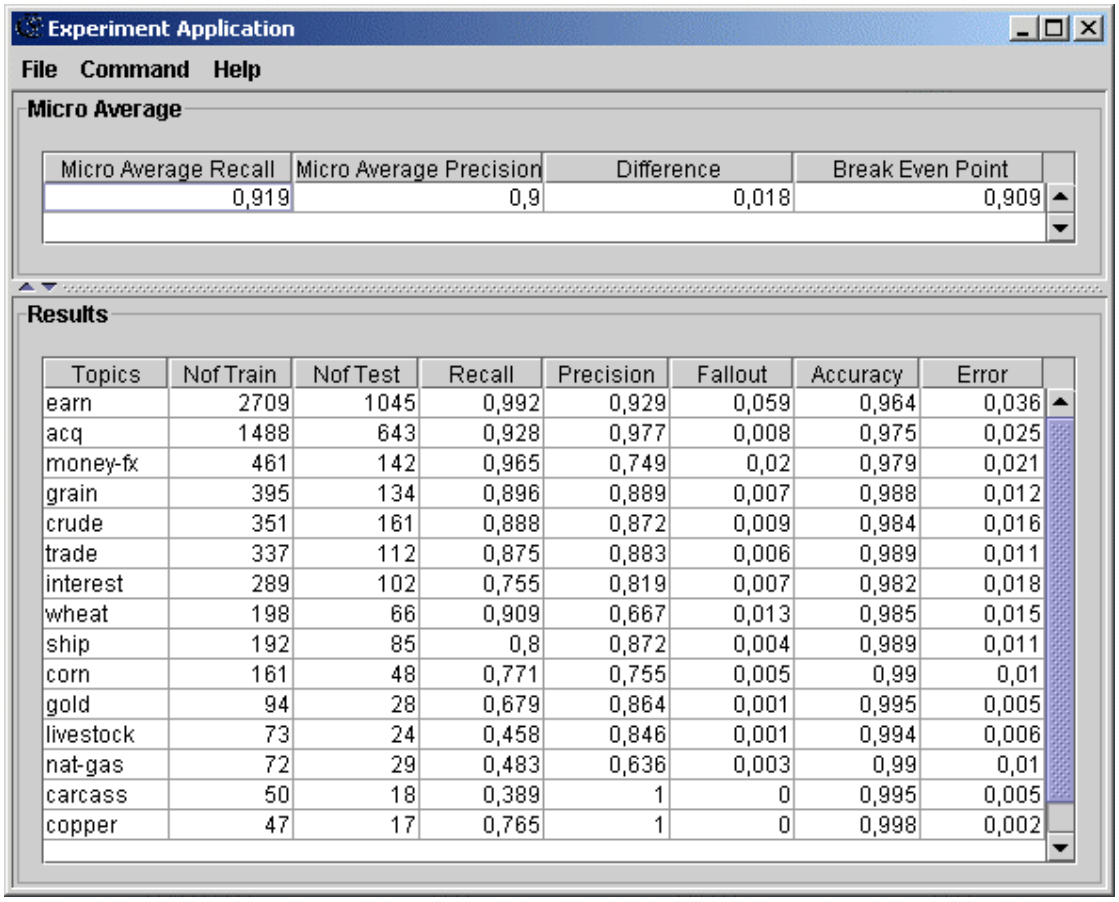
Figur 18. Under viktningsfliken väljs viktningsprincip.



Figur 19. Under metodfliken väljs klassificeringsmetod. I textfältet ”k-Neighbours” kan man testa olika värden för *k*, och under ”Threshold” testa olika tröskelvärden.

Alla test och träningsparametrar är förinställda med rimliga värden. Användaren kan alltså välja vad som skall ställas in. Det minsta som måste göras är att ange träningskollektion, vilket görs under dokumentfliken. Önskar användaren senare under test och träningsförloppet ändra något värde, kan detta göras genom att återigen gå in under ”Options”.

Efter att kalibrerat verktyget är dags att träna och testa klassificeraren. Vilket görs genom att gå in under kommandomenyn. Verktyget tar då automatsikt fram en klassificerare och presenterar utvärderingsresultatet för användaren (se figur 20.).



Topics	Nof Train	Nof Test	Recall	Precision	Fallout	Accuracy	Error
earn	2709	1045	0,992	0,929	0,059	0,964	0,036
acq	1488	643	0,928	0,977	0,008	0,975	0,025
money-fx	461	142	0,965	0,749	0,02	0,979	0,021
grain	395	134	0,896	0,889	0,007	0,988	0,012
crude	351	161	0,888	0,872	0,009	0,984	0,016
trade	337	112	0,875	0,883	0,006	0,989	0,011
interest	289	102	0,755	0,819	0,007	0,982	0,018
wheat	198	66	0,909	0,667	0,013	0,985	0,015
ship	192	85	0,8	0,872	0,004	0,989	0,011
corn	161	48	0,771	0,755	0,005	0,99	0,01
gold	94	28	0,679	0,864	0,001	0,995	0,005
livestock	73	24	0,458	0,846	0,001	0,994	0,006
nat-gas	72	29	0,483	0,636	0,003	0,99	0,01
carcass	50	18	0,389	1	0	0,995	0,005
copper	47	17	0,765	1	0	0,998	0,002

Figur 20. Utvärderingsresultat efter testning.

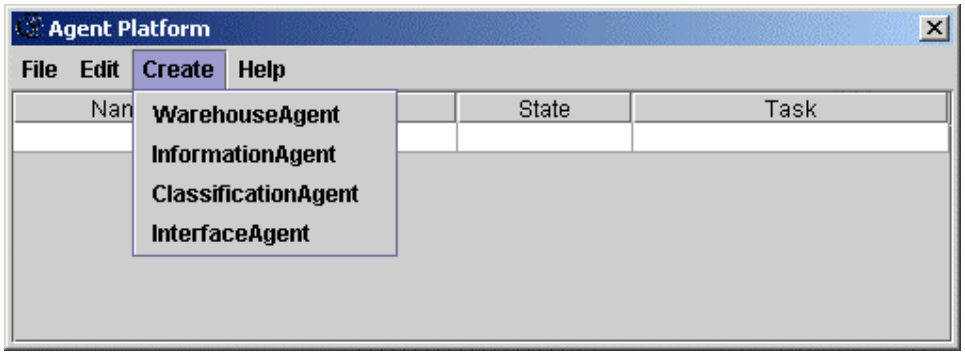
Investeraren får en break-even point på 0.9 vilket han är mycket nöjd med. Han sparar klassificeringsmodulen under filmenyn och stänger sedan träningsverktyget. Hela framtagningsprocessen, som gick på några minuter, är nu klar. Det är dags att börja använda modulen för klassificering av finansiell information.

6.3.4 Registrering av agenter

ERP Text Miner koordinerar en serie samarbetande agenter som förvärvar, klassificerar och förvaltar information från olika källor. För att börja använda systemet, är första steget att registrera agenter. Detta görs genom att gå in under agentmenyn och välja plattformsmenyn (figur 21).

Figur 21. Under menyn “Agents” väljs “Platform” för att registrera agenterna i systemet.

I ERP Text Miner finns en agentplattform vilken är den miljö där de olika agenterna lever och arbetar med varandra. På plattformen kan användaren enkelt registrera agenter för olika uppgifter, starta, stoppa dem eller om önskas ta bort dem från systemet. Figur 22 visar fönstret för agentplattformen.



Figur 22. På agentplattformen registrerar användaren agenterna i systemet.

Från plattformens skapameny väljs de agenter som skall registreras i systemet. När man väljer att skapa en viss agent kommer ett inställningsfönster att poppa up. I detta fönster är det sedan möjligt att skräddarsy agentens beteende genom att ställa in olika värden.

För att få tillgång till eller lägga in dokument i systemet skapas warehouseagenten.

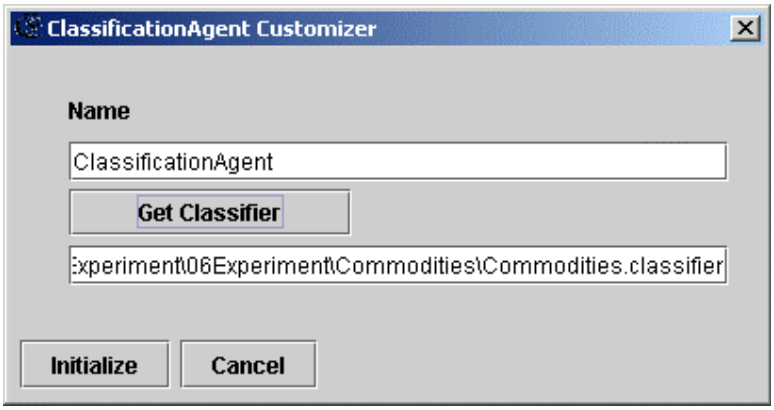
Då i detta fall, investeraren önskar förvärva finansiell information från nyhetsbyrån Reuters, skapas en informationsagent vilken ställs in att koppla upp sig mot denna källa.

För att sedan kunna få struktur i förvärvade dokument skapas en klassificeringsagent. För att agenten skall kunna utföra sin uppgift anges vilken klassificeringsmodul agenten skall arbete med. I detta fall anger investeraren den modul som han tidigare tog fram med träningsverktyget.

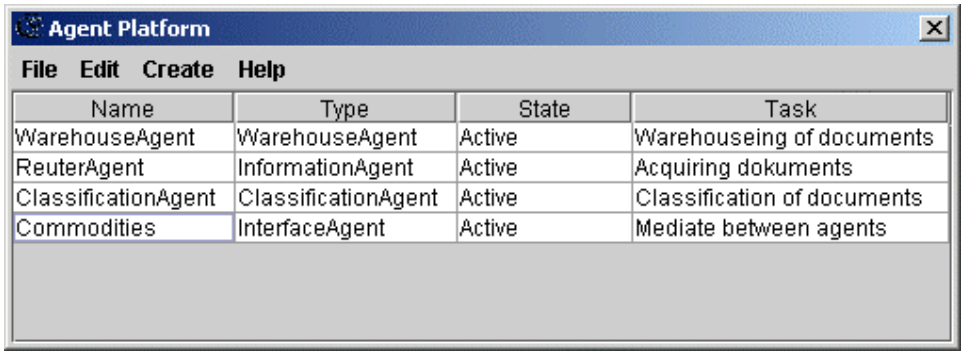
Slutligen skapas en gränssnittsagent, vilken är agenten som användaren direkt kan samverka med, som presenterar klassificerade dokument, och som koordinerar de övriga agenter i systemet.

Efter att aktiverat de olika agenterna stängs plattformsfönstret. Om man senare önskar registrera fler agenter eller ta bort agenter från systemet, görs detta genom att återigen öppna upp fönstret till agentplattformen.

Figur 23, visar registreringen av klassificeringsagenten och figur 24, visar alla registrerade agenter på plattformen efter att de har aktiverats.



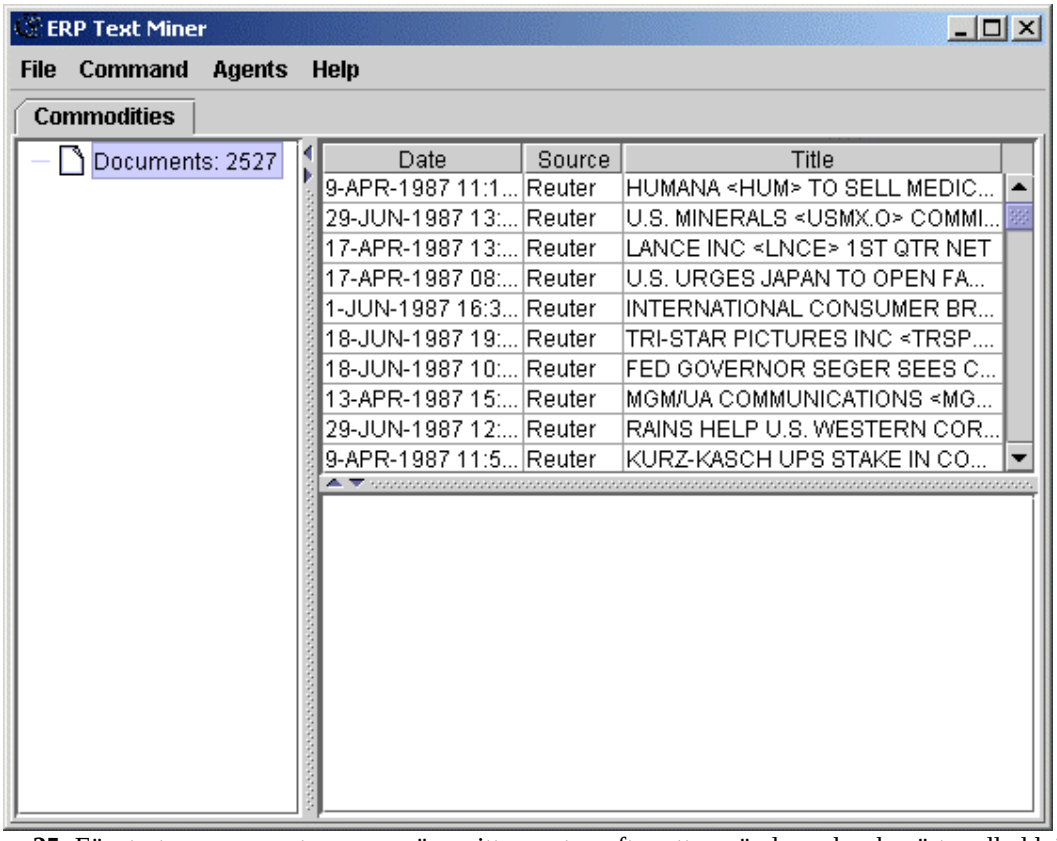
Figur 23. Registrering av klassificeringsagenten. Genom att trycka på knappen ”Get Classifier” går det att ladda den klassificeringsmodul som tidigare tagits fram med träningsverktyget.



Figur 24. Skapade och aktiverade agenter i systemet.

6.3.5 Nedladdning av dokument

När de olika agenterna är aktiverade i systemet börjar de autonomt utföra sina uppgifter och, om så behövs, kooperativ. Informationsagenten kopplar upp sig mot vald informationskälla och sätter igång att förvärva dokument. Dokumenten transformeras till ett för systemet standardiserat format och skickas till warehouseagenten som lägger in dem i dokumentdatabasen. Om nytt dokument redan finns i databasen kommer det äldre dokumentet att ersättas med det nya. Användaren kan nu ladda ner och klassificera dokument i det fönster som gränssnittsagenten ställer till förfogande. Figur 25 återger detta fönster efter att dokument har laddas ner.

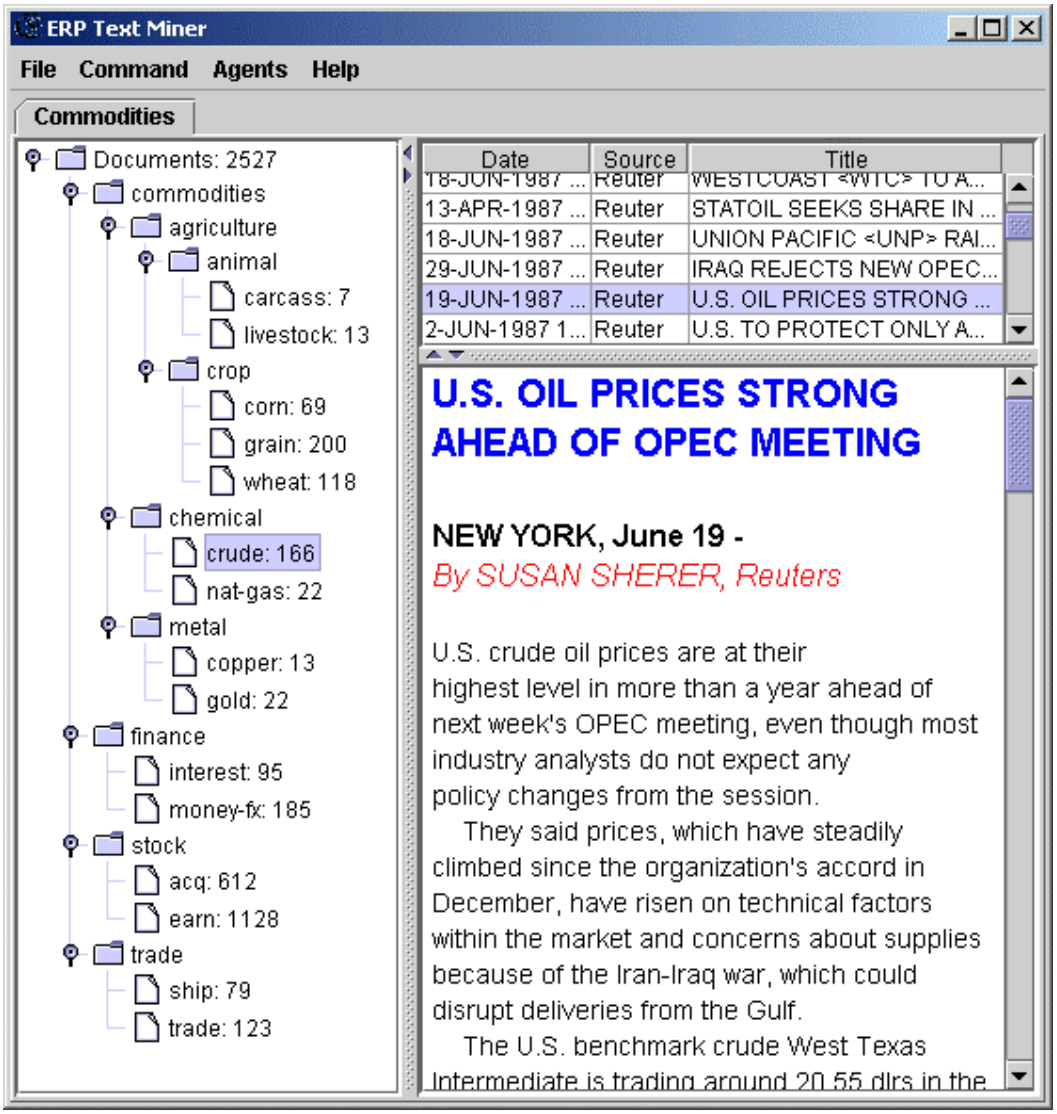


Figur 25. Fönstret som presenteras av gränssnittsagenten efter att användaren har begärt nedladdning. För tillfället finns det 2527 förvärvade dokument i systemet.

6.3.6 Klassificering av dokument

Nu är det möjligt att klassificera dokumenten. För tillfället finns det totalt 2527 dokument tillgängliga i systemet. En siffra som hela tiden stiger när de aktiverade informationsagenter hämtar hem nya dokument. Nu finns det alltså ett behov att snabbt och enkelt kunna hitta den information som för tillfället är relevant. Det som önskas är en struktur, en navigerbar struktur, en struktur där dokument är inordnade i klasser som indikerar informationens innebörd.

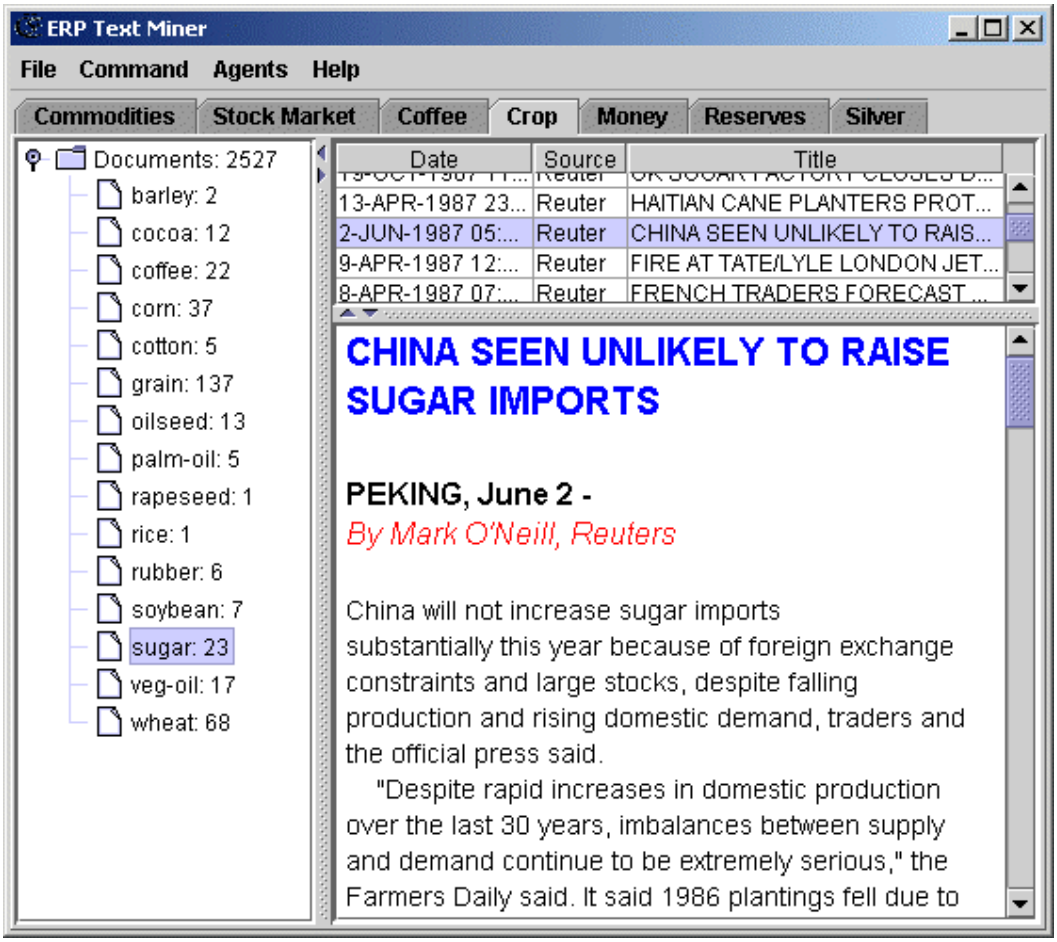
För att klassificera dokumenten går användaren in under kommandomenyn och gör en begäran. Gränssnittsagenten sätter klassificeringsagenten i arbete som då börjar katalogisera dokumenten med den klassificeringsmodul som den försågs med vid registreringen. Klassificerade dokument sänds sedan tillbaka till gränssnittsagenten som presenterar resultatet för användaren. Figur 26 visar dokument klassificerade i en struktur som investeraren i detta scenario önskar sig.



Figur 26. Dokument klassificerade i en navigerbar struktur. De yttersta noderna i trädet anger hur många dokument som finns insorterade i klassen. Den översta noden ”Documents” anger hur många dokument som för tillfället finns tillgängliga i systemet.

6.3.7 Flera dokumentstrukturer samtidigt

Med ERP Text Miner finns möjligheten till flera dokumentstrukturer samtidigt. Med träningsverktyget kan nya klassificeringsmoduler tas fram som strukturerar informationen på önskat sätt. På agentplattformen aktiveras nya klassificeringsagenter och gränssnittsagenter som presenterar resultatet för användaren. Figur 27 visar ERP Text Miner med sju stycken klassificeringsagenter och sju stycken gränssnittsagenter aktiverade.



Figur 27. ERP Text Miner med flera dokumentstrukturer samtidigt.

7 SLUTSATSER OCH AVSLUTANDE DISKUSSION

7.1 Inledning

Detta examensarbete har behandlat utveckling av ERP Text Miner, ett agentbaserat system för förvärv och klassificering av dokument.

Syftet med arbetet var att:

- Förvärva och presentera kunskap om teknikerna automatisk textklassificering och agenter, för att ge förståelse för system som tillämpar dessa tekniker kan utvecklas, utvärderas, kombineras, och i slutändan användas till att snabbare och enklare hitta relevant information.

Utifrån ovanstående syfte ställde jag upp följande frågeställningar:

- Hur utvecklas och utvärderas automatiska textklassificerare, samt vilken prestanda är möjlig att uppnå med en sådan klassificerare?

- Vad kännetecknar en god design för ett system som förvärvar och klassificerar information, samt hur kan man använda ett system som är utvecklat med en sådan design?

För att kunna uppfylla syftet, svara på frågeställningarna, sammanställde jag befintlig kunskap om automatisk textklassificering och agenter. Jag utvecklade en textklassificerare genom att förverkliga typiska utvecklingssteg i processen. Jag utvärderade klassificerarens prestanda med den standardiserade kollektionen Reuters-21578, samt jämförde resultaten med tidigare publicerade resultat där denna kollektion har använts. Jag designade och utvecklade ERP Text Miner, ett agentbaserat system för förvärv och klassificering av dokument. Och slutligen demonstrerade jag systemets användning med ett scenario.

I avsnitten 7.2, 7.3, 7,4 och 7.5, sammanfattas och diskuteras vad jag kom fram till, och i avsnitt 7.6 ges förslag till fortsatt arbetet.

7.2 Utveckling och utvärdering av automatiska textklassificerare

Hur utvecklas och utvärderas en automatisk textklassificerare?

Under `80-talet utvecklades automatiska textklassificeringssystem manuellt, genom framställning av expertsystem som klassificerade dokument med regler. Dessa system var ofta effektiva med samtidigt dyra att utveckla och underhålla beroende på den så kallade kunskaps-förvärvs-flaskhalsen

Numera har maskininlärning blivit den dominerande metoden att utveckla automatiska system för textklassificering. Vid maskininlärning utvecklas ett program som automatisk tar fram en klassificerare. Detta görs genom att programmet ”observerar” karaktäristiken hos en uppsättning dokument som tidigare manuellt har klassificeras av en expert. Utifrån denna förvärvade kunskap kan programmet sedan bedöma om ett nytt dokument skall klassificeras under olika kategorier eller inte.

Det finns ett omfattande arbete rörande maskininlärning från text. En typisk textklassificeringsprocess består av följande steg: *förbehandling; representation; viktning; reducering av dimensioner; val av maskininlärningsmetod (klassificering); samt utvärdering av prestanda.*

Under *förbehandlingen* tas de nyckelord fram som man senare indexerar med. Den lexikaliska analysen identifierar själva orden i texten. Med stoppordsborttagning avlägsnas frekventa

informationsfattiga ord. Genom att omvandla orden till grundform reduceras varianter av samma rotord för ett vanligt koncept. Fördelen med dessa steg är att indexstrukturen kan reduceras betydligt vilket senare leder till bättre klassificeringsförmåga.

Efter att tagit fram indextermer *representeras* dessa sedan. Vanligt är att man använder den statistiska modellen vektor-rymd-modellen, där textens termer viktas och motsvarar en vektor i en rymd. Detta är en så kallad bag-of-words representation vilket innebär att man inte tar någon hänsyn till ordens struktur i original texten. Vektorerna inordnas sedan i term-dokument-matris. I matrisen motsvarar varje kolumn en term, varje rad ett dokument, och varje matrisingång motsvarar vikten hos termen i dokumentet.

Det finns olika sätt att beräkna *vikterna* hos termerna. Exempel på olika viktningsprinciper är: boolesk-viktning, termfrekvens-viktning, tfxidf-viktning och ltc-viktning. De flesta sätt bygger på två empiriska observationer: ju fler gånger en term förkommer i ett dokument, ju mer relevant är det rörande ämnet i dokumentet; ju fler gånger en term förekommer i alla dokument i kollektionen, sämre urskiljningskraft har det mellan dokumenten.

Trots genomförd förbehandling av dokumenten kan antalet indextermer fortfarande vara mycket stort. Det finns en dimension för varje unik term, vilket kan bli tusentals. Därför finns behovet att *reducera antalet dimensioner* för att på så sätt öka systemets klassificeringsförmåga. Ibland genomförs en reparameterisering. Latent Semantic Indexing är ett exempel på en sådan metod. När reparameterisering är olämplig kan i stället en feature selection genomföras. Vid feature selection avlägsnas de termer som har låg informationsvärde för klassificering. Det finns minst fem metoder för feature selection: Document Frequency Thresholding, Information Gain, X^2, Mutual Information, och Term Strength. Experiment har visats att de tre förstnämnda ger bäst resultat.

Det finns många olika *maskininlärningsmetoder* för textklassificering. Exempel är Support Vector Machines, k-Nearest Neighbor, Ripper, Decision Trees, Rocchio och naive Bayes. I detta arbete har jag presenterat och testat metoderna k-Nearest Neighbor och naive Bayes. k-Nearest Neighbor fungerar enligt ett exempelbaserat tillvägagångssätt, där dokumentet *d* som skall klassificeras används som en fråga mot träningsdokumenten. De klasser hos träningsdokumenten som genererar högst klassificeringsstatusvärden anses som lovande kandidater att klassificera *d*. Med ett tröskelvärde fattas sedan det slutliga klassificeringsbeslutet. Naive Bayes bygger på Bayes teorem för betingad sannolikhet. Genom att fastställa förekomstfrekvensen hos varje enskild term sammanfaller med dokumentens klasstilldelning kan sannolikheten för ett dokuments klasstillhörighet beräknas.

Vid *utvärdering* är det vanligt att klassificeringsuppgiften bryts ner i binärt åtskilda klassificeringsproblem. Fyra olika kvantiteter fastställs huruvida ett dokument tillhör en klass eller inte. Från dessa kvantiteter definieras sedan prestandamåtten: recall, precision, fallout, accuracy och error. Ofta måste man kompromissa mellan recall och precision, om dessa mått trimmas så att de får lika värden kallas detta för breakeven punkten hos systemet. Genomsnittsprestanda över många olika klasser kan mätas med macroaverage och microaverage. Microaverage breakeven point är ett vanligt mått för att jämföra klassificera med varandra.

7.3 Prestanda hos en automatisk textklassificerare

Vilken prestanda är möjlig att uppnå med en automatisk textklassificerare?

För att utvärdera prestanda används vanligtvis genomsnittsmåttet microaverage breakeven point, vilket diskuterades ovan. Ytterligare, för att objektivt kunna jämföra olika resultat använder man en standardiserad kollektion för träning och testning. En mycket vanlig sådan kollektion är Reuters-21578. Alltså, för att kunna svara på denna fråga genomförde jag en serie experiment med denna

kollektion. Dom resultat som jag erhöll jämförde jag sedan med andra forskares resultat där samma kollektion användes.

Jag delade upp kollektionen enligt ModApte uppdelningen. En uppdelning som ledde till 7,068 dokument för träning, 2,745 dokument för testning samt 89 stycken klasser.

Under förbehandlingen tog jag bort alla märkord och annan metainformation. Jag avlägsnade stoppord med hjälp av en lista med 887 stycken stoppord. Resterade ord omvandlades sedan till grundform med hjälp av stemmingsalgoritmen Porters stemmer. Förbehandlingen resulterade i 15,291 stycken indextermer.

För representation använde jag vektor-rymd-modellen, där jag organiserade framtagna vektorer i ett inverterad index.

För att reducera dimensioner prövade jag två metoder för feature selection: Document Frequency Thresholding (DFT), samt Mutual Information (MI). Jag fann att båda metoder genererade ganska likvärdiga resultat men att DFT var bättre då den var enklare att implementera och använda. Vid DFT genomförde jag en aggressiv dimensionsreducering. Klassificerarens recall var stabil oavsett antalet reducerade dimensioner, men dess precision och därmed systemets breakeven point förbättrades ju mer jag reducerade ner vokabulären.

Jag experimenterade med två olika metoder för termviktning: ltc och tfxidf. Jag fann att ltc genererade något bättre breakeven än tfxidf.

För klassificering valde jag k-Nearest Neighbor som maskininlärningsmetod, en metod som tidigare har visat mycket god prestanda för textklassificeringsuppgifter. Jag fann att prestanda för min kNN var relativt stabil för stora värden på k. De resultat som återges här generades när k var lika med 45.

Vid ltc-viktning reducerade jag alla termer vars dokumentfrekvens var under 11, vilket gav 2,783 unika termer. Med tröskelvärdet 0,3 hittade jag klassificerarens breakeven point, vilken var 0,81. Detta var det bästa resultat som jag uppnådde med Reuters-21578.

Vid tfxidf-viktning reducerade jag alla termer vars dokumentfrekvens var under 5, vilket gav 4,495 unika termer. Med tröskelvärdet 0,3 hittade jag klassificerarens breakeven point, vilken var 0,80.

Jag jämförde mitt resultat (0,81) med andra rapporterade resultat i litteraturen och drog följande slutsatser:

- State-of-the-art på Reuters-21578 ligger runt breakeven 0,80-0,85 med kNN.
- kNN tycks vara en av de bästa textklassificeringsmetoderna.
- Resultatet för min kNN är helt i klass med state-of-the-art.
- ltc-viktning är en viktningsprincip som fungera bra, trots att det finns ännu mer sofistikerade viktningsprinciper.
- Val av dimensionsreducerande teknik tycks vara kritiskt för att ytterligare kunna öka prestanda.
- Maskininlärning tycks vara en mycket lämplig teknik för att utveckla system för automatisk textklassificering.

7.4 Design av ERP Text Miner

Vad kännetecknar en god design för ett system som förvärvar och klassificerar information?

En god design har inga väsentliga svagheter. Att utvärdera hur väl detta uppfylls kan göras genom granskning, experiment, och genom att diskutera kring de kriterier som låg till grund för designen[107].

ERP Text Miner (ETM) är uppbyggt med en serie agenter som autonomt och kooperativt förvärvar, förvaltar och klassificerar information för sina användare. Nedan förs en diskussion utifrån de kriterier som låg till grund för designen och hur väl de uppfylls.

Användbart
Med *användbarheten* avses i vilken utsträckning systemet tillfredsställer användarnas behov. ETM har hög användbarhet. ETM syftar till att tillfredställa behovet att snabbare och enklare hitta relevant information. Detta behov tillfredsställer ETM väl. Med träningsverktyget kan man på några minuter ta fram en ny klassificerare, med informationsagenterna kan användaren styra var dokument skall hämtas, på några sekunder kan dessa dokument struktureras upp i navigerbara trädhierarkier. Med dessa strukturer är det sedan enkelt att klicka och bläddra sig fram till relevant information.

Flexibelt
Flexibilitet är ett mått på mödan eller kostnaden att ändra i systemet när det är igång. Tekniken för att utveckla flexibla system är modularisering med hjälp av inkapsling. ETM har en hög flexibilitet. Kärnan i ETM är den agentplattform som de olika agenterna lever, arbetar och samverkar på. Agentplattformen ger en mycket flexibel bas där man kan aktiverar olika agenter, ställa in dem för olika uppgifter, starta dem, stoppa dem, och om så önskas ta bort dem från systemet. Agenterna kapslar in alla utvecklingstekniska detaljer och användarna behöver endast samverka med agenterna utifrån dom tjänster som ställs till förfogande.

Begripligt
Med *begriplighet* avses hur lätt det är att skaffa sig överblick över och förstå systemet. Det principiella verktyget för att uppnå begriplighet är abstraktion. ETM har hög begriplighet. ETM är uppbyggt med agenter, och dessa agenter är ett utmärkt redskap för att hantera komplexiteten vid systemutveckling. Dom ger en naturlig abstraktion för att göra systemet modulärt, dom ger ett bättre sätt att konceptualisera, designa och implementera systemet. Naturliga omvärldsentiteter och deras interaktioner kan direkt mappas till problemlösande agenter med egna resurser och expertis.

Portabelt
Portabelt är ett mått på mödan att flytta systemet till andra tekniska plattformar. ETM har hög portabilitet. ETM är helt utvecklat i Java vars stora styrka är just portabilitet. Källkoden kompileras till en mellanform kallad bytecode vilken lagras i klassfiler. Denna bytecode kan sedan exekveras på alla datorsystem som kan installera Java Virtual Machine, vilken är den tolk översätter bytecode till den underliggande tekniska plattformen.

Integrerbart
Med *integrerbart* avses hur lätt det är att sammankoppla systemet med andra system. ETM har hög integrerbarhet. Informationsagenten i ETM är den agenttyp som förvärvar information från olika informationskällor. Om behovet uppstår att koppla upp och hämta information från nya informationskällor kan detta enkelt lösas, genom att utveckla nya informationsagenter och plugga in dessa i ETM. I informationsagenten implementeras då det systemgränssnitt som behövs för interaktionen.

[107] Mathiassen, Munk-Madsen, Nielsen & Stage: ”Objektorienterad analys och design”

Hög samhörighet och låg koppling
System bör ha hög samhörighet och låg koppling. Hög *samhörighet* är ett uttryck för hur väl varje enskild agent hänger samman. Hög samhörighet betyder att agenten framstår som en helhet, som har väsentliga relationer mellan sina delar. Namngivning av agenter är en väsentlig indikator på graden på samhörighet. Vid hög samhörighet är det lätt att hitta ett precist och heltäckande namn för agenten. ETM har hög samhörighet. Alla agenter har enkla och beskrivande namn vilka inrymmer deras funktioner utan att konjunktionen ”och” behöver användas.

Koppling är ett uttryck för hur nära två agenter hänger samman. Låg koppling innebär att om den ena agenten ändras så berörs inte den andra agenten av detta. En design med få länkar mellan agenterna är mindre kopplat och därför mer önskvärt. ETM har låg koppling. Det existerar inte några direkta länkar mellan agenterna utan interaktionen mellan agenterna görs genom utbyte av meddelanden.

Alltså utifrån genomförda granskningar, experiment och diskussionen ovan drar jag slutsatsen att ETM kännetecknar en god design för ett system som förvärvar och klassificerar information. Designen har inga väsentliga svagheter. Systemet är användbart, flexibelt, begripligt, portabelt, integrerbart, systemet har en hög samhörighet och en låg koppling mellan sina delar.

7.5 Användningen av ERP Text Miner

Hur kan man använda ett system som är utvecklat med vald design?

ERP Text Miner (ETM) stödjer hela textklassificeringsprocessen, från framtagning och utvärdering av textklassificeringsmoduler, till tillämpningen av dessa moduler i ett agentbaserat system.

ETM erbjuder dels ett träningsverktyg där användare kan ta fram och uttesta textklassificeringsmoduler för valfria dokumentkollektioner, dels en agentplattform där användaren kan aktivera olika agenter som förvärvar och klassificerar dokument.

För att utveckla en klassificeringsmodul sätter användaren ihop en dokumentkollektion där de olika ingående klasserna definieras. Träningsverktyget kalibreras sedan genom att ange lämpliga val för de olika stegen i textklassificeringsprocessen. När klassificeraren är testad och användaren är nöjd med resultatet sparas modulen, vilken då är klar att används för klassificering av nya dokument.

För att börja använda ETM, öppnar användaren agentplattformen. Agentplattformen är den miljö där de olika agenterna lever och arbetar med varandra. På plattformen kan användaren enkelt registrera agenter, starta och stoppa dem eller om så önskas ta bort dem från systemet.

När de olika agenterna är aktiverade i systemet börjar de autonomt, och om så behövs kooperativt, att utföra sina uppgifter. Informationsagenterna förvärvar dokument som förvaltas av warehouseagenten. Användaren kan nu klassificera dokument genom att göra en begäran hos gränssnittsagenten. Gränssnittsagenten aktiverar berörd klassificeringsagent som klassificerar dokumenten och sänder tillbaka dessa till gränssnittsagenten vilken då uppdaterar användargränssnittet.

Med ETM kan användaren också arbeta med flera dokumentstrukturer samtidigt. Med träningsverktyget kan nya klassificeringsmoduler tas fram. På agentplattformen kan nya agenter aktiveras vilka presenterar sina resultat för användaren.

7.6 Förslag till fortsatt arbete

Användargränssnittet till ERP Text Miner är designat utifrån det jag själv fann mest användarvänligt. Förslag till fortsatt arbete är att genomföra mer omfattande användarstudier där både gränssnittet och systemet som helhet kunde utvärderas med större inblandning av potentiella användare.

När de olika agenterna i ERP Text Miner kommunicerar, görs detta genom direkt kommunikation, i med att alla agenterna pratar samma språk och syntaxen i meddelandena är relativt enkel. För att kunna utveckla mer sofistikerade meningsutbyten skulle det vara intressant att experimentera med en ontologi och ett gemensamt språk som KQML, för att utbyta information och kunskap i systemet. Förslag på fortsatt arbete kunde också vara att införa en tolk eller facilitator som agenterna kunde kommunicera genom.

När jag jämförde min klassificerare med andra rapporterade arbeten i litteraturen, drog jag slutsatsen att val av dimensionsreducerande teknik tycks vara kritiskt för att ytterligare kunna öka klassificeringsprestanda. Förslag till fortsatt arbete kunde därför vara att experimentera med andra metoder för feature selection. Framförallt skulle det vara intressant att testa Information Gain och X^2.

En del arbeten inom textklassificering utvidgar bag-of-words representationen genom att använda ordsekvenser istället för ensamstående ord. Dessa arbeten föreslår att ordpar som termer i bag-of-words representationen kan förbättra klassificeringsprestanda för kortare dokument. I med att många dokument är ganska korta skulle det därför vara intressant att undersöka denna representationsform närmare.

8 REFERESNER

8.1 Tryckta källor

Apté, C. Damerau, F., & Weiss, S. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233-251.

Baeza-Yates, R, & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley Longman

Biebricher, P., Fuhr, N., Lustig, G., & Schwantner, M. (1988). The automatic indexing system AIR/PHYS. From research to application. In *Proceedings of SIGIR-88, 11th ACM International Conference on Research and Development in Information Retrieval*, pages 333-342, Genoble, FR.

Bigus, J.P., Bigus J. (2001). *Constructing Intelligent Agents Using Java*. Second edition. Wiley.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation 2:14-23.

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1994). Automatic Query Expansion Using SMART: TREC 3. In *Proceedings of 3rd Text Retrieval Concerence*, NIST.

Castelfranchi, C. (1995), "Guarantees for autonomy in cognitive agent architecture", in Wooldridge, M.J. and Jennings, N.R. (Eds), *Intelligent Agents – proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Amsterdam, 8-9 August, Springer-Verlag, pp. 56-70.

Chen, H. & Dumais, S.T. (2000). Bringing order to the web: Automatically categorizing search results. In *Proceedings of CHI'00, Human Factors in Computing Systems*.

D'Aloisi, D. and Giannini, V. (1995), "The Info Agent: An Interface for Supporting Users in Intelligent Retrieval", In Proceedings of the ERCIM Workshop "Towards User Interfaces for All: Current Trend and Future Efforts", pages 143-145.

Dahlbom, B.(1996). The New Informatics. *Scandinavian Journal of Information Systems*, Vol.8, No.2.

Dahlbom, B. & Mathiassen, L. (1993). *Computers in Context: The philosophy and practice of systems design*. MA: Blackwell, Cambridge.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*. Vol 41, No. 6. pp. 391-407.

Dumais, S., Platt, J., Hekerman, D. & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98*.

Etzioni, O. & Weld, D. (1995), "Intelligent agents on the Internet: fact, fiction and forecast", IEEE Expert, August.

Franklin, S. and Graesser, (1996), "Is it an agent, or just a program? A taxonomy for autonomous agents", *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag.

Fuhr, N., Hartman, S., Knorz, G., Lustig, G., Schwantner, M., & Tzeras K. (1991). AIR/X – a rule-based multistage indexing system for large subject fields. In *Proceedings of RIAO´91, 3rd*

International Conference "Recherche d´Information Assistee par Ordinateur", pages 606-623, Barcelona, ES.

Jennings, N.R. and Wooldridge, M.(1998). "Applications of Intelligent Agents". In: Jennings NR, and Wooldridge MJ (Eds). *Agent Technology Foundations, Applications, and Markets*. Springer-Verlag.

Herou, E. (2002). Ett träningsbart verktyg för att klassificera nyhetstexter. *Examensarbete inom datalogi NADA, KTH*.

Hersh, W., Buckley, C., Leone, T., & Hickman, D. (1994). OHSUMED: an interactive retrieval evaluation and new large text collection for research. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 192-201, Dublin, IE.

Ingham, J. (1999). "What is an Agent?" *Technical Report*, #6/99, from Center for Software Maintenance University of Durham.

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*.

King, J.A. (1995). "Intelligent Agents: Bring Good Things to Life", *AI Expert*, Feb. & Mar.

Klusch, M. (2001). "Information agent technology for the Internet: A survey". *Data & Knowledge Engineering*. 36 pp. 337-372.

Maes, P. (1994), "Agents that reduce work and information overload", *Communications of the ACM*, Vol. 71 No. 7, July, pp. 31-40.

Mathiassen, L., Munk-Madsen, A., Nielsen, P-A. & Stage, J. (1998). *Objektorienterad analys och design*. Studentlitteratur, Lund.

Miles, M.B. & Huberman, M.A. (1994). *Quality data analysis*. Sage Publications.

Mitchell, T., Caruana, R., Freitag, D., McDermott and Zabowski, D. (1994). "Experience with a learning personal assistant", *Communications of the ACM*, Vol. 17 No. 7, pp. 81-91.

Patel, R. & Davidson, B. (1994). *Forskningsmetodikens grunder – att planera, genomföra och rapportera en undersökning* (2:a upplagan). Lund: Studentlitteratur

Porter, M.F. (1980). An Algorithm for Suffix Stripping. *Program* 14 (3), pp. 130-137, July

Sebastiani, F. (1999). A tutorial on automated text categorisation. In *Analia Amandi and Ricardo Zunino, editors, Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 7--35, Buenos Aires, AR, 1999.

Shütze, H., Hull, D.A., & Pedersen, J.O. (1995). A comparison of classifiers and document representations for the routing problem. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229-237, Seattle US.

Svenning, C. (1999). *Metodboken. En bok om samhällsvetenskaplig metod och metodutveckling.* Lorentz.

Weiss, S.M., Apté, C., & Damerau, F.J. (1999). Maximizing Text-Mining Performance. *IEEE Intelligent Systems*.

Wooldridge, M.J. and Jennings, N.R. (1995), “Intelligent agents: theory and practice”, *The Knowledge Engineering Review*, Vol. 10 No. 2, pp. 115-52.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Journal of information retrieval.* 1-2(1):69-90.

Yang, Y. & Liu, X. (1999). A re-examination of text categorisation methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, Berkeley, US.*

Yang, Y., & Pedersen, J.P. (1997). Feature selection in statistical learning of text categorization. In the *14th Int. Conference on Machine Learning*, pp. 412-420.

Yang, Y., & Wilbur, W.J. (1996) Using corpus statistics to remove redundant words in text categorization. In *Journal of the American Society for Information Science*.

8.2 Internet källor

Aas K, Eikvil, L. (1999). *Text Categorisation: a survey*.
http://www.nr.no/documents/samba/research_areas/BAMG/Publications/tm_survey.ps
Senast besökt 2003-06-30

Bennet, P. (2002). Introduction to Text Categorization.
www.andrew.cmu.edu/course/20-760/notes/ intro-textclassification.ppt
Senast besökt 2003-11-26

Carlberger J., Dalianis H., Hassel, M., Knutsson, O. (2001). *Improving precision in information retreival for Swedish using stemming*.
http://www.nada.kth.se/~xmartin/papers/Stemming_NODALIDA01.pdf
Senast besökt 2003-08-05

Parunak HVD, (1998). Practical and Industrial Applications of Agent-Based Systems. *Industrial Technology Institute*.
Tillgänglig vid: http://www.cs.umbc.edu/agents/introduktion

Reuters-21578 testkollektion
http://www.daviddlewis.com/resources/testcollections/reuters21578/
Senast besökt 2003-06-30

APPENDIX 1 – REUTERS-21578

Reuters-21578 består av artiklar som var publicerade vid Reuters nyhetsbyrå under året 1987. Anställda vid Reuters Ltd., och Carnegie Group Inc., samlade in artiklarna och indexerade dem under olika kategorier.

År 1990, tillgängliggjordes dokumenten av Reuters och CGI till forskningsavdelningen Information Retrieval Laboratory vid avdelningen för Computer and Information Science, vid University of Massachusetts at Amherst, för forskningsändamål. Då formaterades dokumenten och associerade datafiler togs fram av bland annat av David D. Lewis och Stephen Harding.

För att möjliggöra att publicerade resultat skulle kunna jämföras mellan olika studier, genomförde Steve Finch och David D. Lewis år 1996 en upprensning samt upptagning av samlingen med SGML. Numera består samlingen av 21578 dokument, därmed namnet.

Reuters-21578, och den distribution som i skrivande stund finns tillgänglig, är version 1.0, som går under namnet "Reuters-21578, Distribution 1.0".

Filformat

Reuters-21578 test korpus består av 22 stycken SGML data filer, en SGML DTD fil som beskriver filformatet, och sex filer som beskriver de kategorier som används för att indexera texten. Även ytterligare filer ingår, vilka inkluderats av olika forskar. Dessa filer är inte är en del av själva test-samlingen, men de kan utgöra ytterligare en resurs om dem används.

De första 21 stycken filerna (reut2-000.sgm till reut2-020.sgm) innehåller 1000 dokument vardera, medan den sista filen (reut2-021.sgm) innehåller 578 dokument.

Filerna är i SGML format. Och härs görs endast en informell beskrivning hur olika SGML taggar används för att dela upp varje fil, och varje dokument, i olika sektioner.

Var och en av filerna börjar med en dokument-fil-deklaration:

<!DOCTYPE lewis SYSTEM ”lewis.dtd”>

DTD filen ”lewis.dtd” är inkluderad i distributionen. De taggar som följer dokument-typ-deklarations taggen är olika individuella Reuters artiklar märkta med SGML taggar vilka beskrivs nedan.

Varje artikel börjar med en ”öppnings tagg” av formen:

<REUTERS TOPICS=?? LEWISSPLIT=?? CGISPLIT=?? OLDID=?? NEWID=??>

där ?? är ifyllda i det sammanhang där attributen förekommer. Varje artikel avslutas med en ”stängningstagg” av formen:

</REUTERS>

Varje REUTERS tagg anger en uttrycklig specifikation hos de fem attributen: TOPICS, LEWISSPLIT, CGISPLIT, OLDID, and NEWID. Dessa attribut är avsedda för identifiering och gruppering av de olika dokumenten. Värdena hos dessa attribut bestämmer hur dokumenten kan delas upp i delmängder för träning och testning.

Split

Vid experiment med Reuters-21578 är det möjligt att dela upp textsamlingen i olika på förhand givna partitioner, eller split, och på så sätt få olika standardiserade konfigurationer av dokumenten. Detta möjliggör jämförelse mellan andra presenterade resultat.

Den medföljande dokumentationen anger tre olika typer av split: Modified Lewis, (”ModLewis”) split, Modified Apte (”ModApte”) split, samt, Modified Hayes (”ModHeys”) split.

I detta arbete används ModApte, vilken är den partitionering som mest frekvent förekommer i litteraturen. ModApte split fås utifrån följande val av parametrar:

Tränings Set	(9,603 dokument):	LEWISSPLIT=”TRAIN”;	TOPICS=”YES”
Test Set	(3,299 dokument):	LEWISSPLIT=”TEST”;	TOPICS=”YES”
Används ej	(8,676 dokument):	LEWISSPLIT=”NOT-USED;	TOPICS=”YES”
		Eller TOPICS=”NO”	
		Eller TOPICS=”BYPASS”	

Attributen CGISPLIT, OLDID, och NEWID ignorerades i detta arbete.

Interna taggar

På samma sätt som taggarna <REUTERS> och </REUTERS> används att avgränsa dokument inom filerna, används andra taggar för att avgränsa de olika delarna inom ett dokument. Dessa taggar är <DATE>, <MKNOTE>, <TOPICS>, <PLACECES>, <PEOPLE>, <ORGS>, <EXCHANGE>, <COMPANIES>, <UNKNOWN>, <TEXT>. Av dessa taggar användes här endast <TOPICS> och <TEXT>.

Taggarna <TOPICS>, </TOPICS> kapslar in en lista med olika kategorier, om det finns några kategorier för dokumentet. Om kategorier är angivna är var och en inkapslade i taggarna <D> och </D>.

Taggarna <TEXT>, </TEXT> kapslar in själva texten i dokumentet. En del kontrolltecken eller annan typ av ”skräp-” tecken kan också vara inkluderad. Den ursprungliga blankstegs strukturen i texten har bevarats i ursprungligt skick. Följande taggar avskiljer olika element inom TEXT elementet: <AUTOR>, </AUTOR>, <DATELINE>, </DATELINE>, <TITLE>, </TITLE>, <BODY>, </BODY>. Av dessa taggar användes här de två senast uppräknade (titeln extraherades ut, men användes inte för kategorisering), vilka bäddar in huvuddelen av texten i dokumentet.

Exempel

Nedan återger ett exempel på ett dokument:

```
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="5544" NEWID="1">
<DATE>26-FEB-1987 15:01:01.79</DATE>
<TOPICS><D>cocoa</D></TOPICS>
<PLACES><D>el-salvador</D><D>usa</D><D>uruguay</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
```

<UNKNOWN>
&#5;&#5;&#5;C T
&#22;&#22;&#1;f0704&#31;reute
u f BC-BAHIA-COCOA-REVIEW 02-26 0105</UNKNOWN>
<TEXT>&#2;
<TITLE>BAHIA COCOA REVIEW</TITLE>
<DATELINE> SALVADOR, Feb 26 - </DATELINE><BODY>Showers continued throughout the week in the Bahia cocoa zone, alleviating the drought since early January and improving prospects for the coming temporao, although normal humidity levels have not been restored, Comissaria Smith said in its weekly review.

The dry period means the temporao will be late this year.

Arrivals for the week ended February 22 were 155,221 bags of 60 kilos making a cumulative total for the season of 5.93 mln against 5.81 at the same stage last year. Again it seems that cocoa delivered earlier on consignment was included in the arrivals figures.

.

.

Final figures for the period to February 28 are expected to be published by the Brazilian Cocoa Trade Commission after carnival which ends midday on February 27.

Reuter
&#3;</BODY></TEXT>
</REUTERS>

Klasser

Klasserna, eller TOPICS, i Reuters-21578 är av den ekonomiska typen. Exempel är *earnings, acquisition, grain, crude, trade,* etc. Denna uppsättning med 135 stycken klasser är den som används i nästan all tidigare forskning med Reutersdata. Nedan återges klasserna:

acq alum austdlr austral barley bfr bop can carcass castor-meal castor-oil castorseed citruspulp cocoa coconut coconut-oil coffee copper copra-cake corn corn-oil cornglutenfeed cotton cotton-meal cotton-oil cottonseed cpi cpu crude cruzado dfl dkr dlr dmk drachma earn escudo f-cattle ffr fishmeal flaxseed fuel gas gnp gold grain groundnut groundnut-meal groundnut-oil heat hk hog housing income instal-debt interest inventories ipi iron-steel jet jobs l-cattle lead lei lin-meal lin-oil linseed lit livestock lumber lupin meal-feed mexpeso money-fx money-supply naphtha nat-gas nickel nkr nzdlr oat oilseed orange palladium palm-meal palm-oil palmkernel peseta pet-chem platinum plywood pork-belly potato propane rand rape-meal rape-oil rapeseed red-bean reserves retail rice ringgit rubber rupiah rye saudriyal sfr ship silk silver singdlr skr sorghum soy-meal soy-oil soybean stg strategic-metal sugar sun-meal sun-oil sunseed tapioca tea tin trade tung tung-oil veg-oil wheat wool wpi yen zinc

APPENDIX 2 – STOPPORD

Nedanstående lista är de 887 stoppord som används i detta arbete.

10	39	NULL	a	able	about
above	according	accordingly	across	actually	ad
adj	ae	af	after	afterwards	ag
again	against	ai	al	all	allow
allows	almost	alone	along	already	also
although	always	am	among	amongst	an
and	another	any	anybody	anyhow	anyone
anything	anyway	anyways	anywhere	ao	apart
appear	appreciate	appropriate	aq	ar	are
aren	aren't	around	arpa	as	aside
ask	asking	associated	associates	at	au
available	aw	away	awfully	az	b
ba	bb	bd	be	became	because
become	becomes	becoming	been	before	beforehand
begin	beginning	behind	being	believe	below
beside	besides	best	better	between	beyond
bf	bg	bh	bi	billion	bj
bm	bn	bo	both	br	brief
bs	bt	but	buy	bv	bw
by	bz	c	ca	came	can
can't	cannot	cant	caption	cause	causes
cc	cd	certain	certainly	cf	cg
ch	changes	ci	ck	cl	clearly
click	cm	cn	co	co.	com
come	comes	concerning	consequently	consider	considering
contain	containing	contains	copy	corresponding	could
couldn	couldn't	course	cr	cs	cu
currently	cv	cx	cy	cz	d
de	definitely	described	despite	did	didn
didn't	different	dj	dk	dm	do
does	doesn	doesn't	doing	don	don't
done	down	downwards	during	dz	e
each	ec	edu	ee	eg	eh
eight	eighty	either	else	elsewhere	end
ending	enough	entirely	er	es	especially
et	etc	even	ever	every	everybody
everyone	everything	everywhere	ex	exactly	example
except	f	far	few	fi	fifth
fifty	find	first	five	fj	fk
fm	fo	followed	following	follows	for
former	formerly	forth	forty	found	four
fr	free	from	further	furthermore	fx
g	ga	gb	gd	ge	get
gets	getting	gf	gg	gh	gi
given	gives	gl	gm	gmt	gn
go	goes	going	gone	got	gotten
gov	gp	gq	gr	greetings	gs

gt	gu	gw	gy	h	had
happens	hardly	has	hasn	hasn't	have
haven	haven't	having	he	he'd	he'll
he's	hello	help	hence	her	here
here's	hereafter	hereby	herein	hereupon	hers
herself	hi	him	himself	his	hither
hk	hm	hn	home	homepage	hopefully
how	howbeit	however	hr	ht	htm
html	http	hu	hundred	i	i'd
i'll	i'm	i've	i.e.	id	ie
if	ignored	ii	il	im	immediate
in	inasmuch	inc	inc.	indeed	indicate
indicated	indicates	information	inner	insofar	instead
int	into	inward	io	iq	ir
is	isn	isn't	it	it's	its
itself	j	je	jm	jo	join
jp	just	k	ke	keep	keeps
kept	kg	kh	ki	km	kn
know	known	knows	kp	kr	kw
ky	kz	l	la	last	lately
later	latter	latterly	lb	lc	least
less	lest	let	let's	li	like
liked	likely	little	lk	ll	look
looking	looks	lr	ls	lt	ltd
lu	lv	ly	m	ma	made
mainly	make	makes	many	may	maybe
mc	md	me	mean	meantime	meanwhile
merely	mg	mh	microsoft	might	mil
million	miss	mk	ml	mm	mn
mo	more	moreover	most	mostly	mp
mq	mr	mrs	ms	msie	mt
mu	much	must	mv	mw	mx
my	myself	mz	n	na	name
namely	nbsp	nc	nd	ne	near
nearly	necessary	need	needs	neither	net
netscape	never	nevertheless	new	next	nf
ng	ni	nine	ninety	nl	no
nobody	non	none	nonetheless	noone	nor
normally	not	nothing	novel	now	nowhere
np	nr	nu	nz	o	obviously
of	off	often	oh	ok	okay
old	om	on	once	one	one's
ones	only	onto	or	org	other
others	otherwise	ought	our	ours	ourselves
out	outside	over	overall	own	p
pa	page	particular	particularly	pdf	pe
per	perhaps	pf	pg	ph	pk
pl	placed	please	plus	pm	pn
possible	pr	presumably	probably	provides	pt
pw	py	q	qa	que	quite
qv	r	rather	rd	re	really

reasonably	recent	recently	regarding	regardless	regards
relatively	reserved	respectively	right	ring	ro
ru	rw	s	sa	said	same
saw	say	saying	says	sb	sc
sd	se	second	secondly	see	seeing
seem	seemed	seeming	seems	seen	self
selves	sensible	sent	serious	seriously	seven
seventy	several	sg	sh	shall	she
she'd	she'll	she's	should	shouldn	shouldn't
si	since	site	six	sixty	sj
sk	sl	sm	sn	so	some
somebody	somehow	someone	something	sometime	sometimes
somewhat	somewhere	soon	sorry	specified	specify
specifying	sr	st	still	stop	su
sub	such	sup	sure	sv	sy
sz	t	take	taken	taking	tc
td	tell	ten	tends	test	text
tf	tg	th	than	thank	thanks
thanx	that	that'll	that's	thats	the
their	theirs	them	themselves	then	thence
there	there'll	there's	thereafter	thereby	therefore
therein	theres	thereupon	these	they	they'd
they'll	they're	they've	think	third	thirty
this	thorough	thoroughly	those	though	thousand
three	through	throughout	thru	thus	tj
tk	tm	tn	to	together	too
took	toward	towards	tp	tr	tried
tries	trillion	truly	try	trying	tt
tv	tw	twenty	twice	two	tz
u	ua	ug	uk	um	un
under	unfortunately	unless	unlike	unlikely	until
unto	up	upon	us	use	used
useful	uses	using	usually	uucp	uy
uz	v	va	value	various	vc
ve	very	vg	vi	via	viz
vn	vs	vu	w	want	wants
was	wasn	wasn't	way	we	we'd
we'll	we're	we've	web	webpage	website
welcome	well	went	were	weren	weren't
wf	what	what'll	what's	whatever	when
whence	whenever	where	whereafter	whereas	whereby
wherein	whereupon	wherever	whether	which	while
whither	who	who'd	who'll	who's	whoever
whole	whom	whomever	whose	why	will
willing	wish	with	within	without	won
won't	wonder	would	wouldn	wouldn't	writeln
ws	www	x	y	ye	yes
yet	you	you'd	you'll	you're	you've
your	yours	yourself	yourselves	yt	yu
z	za	zero	zm	zr	

APPENDIX 3 – RESULTAT LTC-VIKTNING

Nedanstående tabell återger klassificerarens prestanda vid respektive klass. För att vikta termerna användes här ltc. Värdena är avrundade med tre decimaler. Antalet dokument för träning och testning är 7,068 och 2,745. Antalet testklasser är 89. Stoppordsborttagning samt suffixstripping med Porter stemmer har genomförts. Värdet på k samt tröskelvärdet är 45 och 0,3. Document frequency Thresholding användes för att reducera dimensioner. Alla termer med dokumentfrekvens under 11 avlägsnades, vilket gav 2,783 unika termer. Micro average breakeven var 0,81.

Topic	*Nof train*	*Nof test*	*Recall*	*Precision*	*Fallout*	*Accuray*	*Error*
earn	2709	1045	0,995	0,874	0,088	0,944	0,056
acq	1488	643	0,955	0,936	0,02	0,974	0,026
money-fx	461	142	0,958	0,673	0,025	0,974	0,026
grain	395	134	0,903	0,716	0,018	0,978	0,022
crude	351	161	0,932	0,777	0,017	0,98	0,02
trade	337	112	0,839	0,653	0,019	0,975	0,025
interest	289	102	0,804	0,719	0,012	0,981	0,019
wheat	198	66	0,909	0,6	0,015	0,983	0,017
ship	192	85	0,835	0,816	0,006	0,989	0,011
corn	161	48	0,813	0,629	0,009	0,988	0,012
sugar	118	35	0,743	0,963	0	0,996	0,004
oilseed	117	44	0,477	0,724	0,003	0,989	0,011
coffee	110	27	0,778	0,875	0,001	0,997	0,003
dlr	96	31	0,613	0,452	0,008	0,987	0,013
gold	94	28	0,714	0,714	0,003	0,994	0,006
gnp	92	34	0,941	0,8	0,003	0,996	0,004
money-supply	87	23	0,652	0,833	0,001	0,996	0,004
veg-oil	86	37	0,541	0,952	0	0,993	0,007
livestock	73	24	0,542	0,722	0,002	0,994	0,006
soybean	73	29	0,345	0,833	0,001	0,992	0,008
nat-gas	72	29	0,517	0,6	0,004	0,991	0,009
bop	62	28	0,536	0,517	0,005	0,99	0,01
cpi	60	26	0,538	0,538	0,004	0,991	0,009
cocoa	50	15	0,867	0,765	0,001	0,998	0,002
carcass	50	18	0,611	0,786	0,001	0,996	0,004
reserves	48	14	0,357	1	0	0,997	0,003
copper	47	17	0,824	1	0	0,999	0,001
jobs	41	18	0,5	1	0	0,997	0,003
iron-steel	40	12	0,417	0,833	0	0,997	0,003
cotton	38	20	0,35	1	0	0,995	0,005
yen	36	12	0,167	0,667	0	0,996	0,004
rubber	35	12	0,583	0,875	0	0,998	0,002
ipi	35	10	0,3	1	0	0,997	0,003
rice	35	24	0,167	1	0	0,993	0,007
barley	33	12	0,417	1	0	0,997	0,003
alum	33	20	0,45	1	0	0,996	0,004
gas	31	14	0,286	1	0	0,996	0,004
meal-feed	30	18	0,056	1	0	0,994	0,006
palm-oil	29	10	0,7	1	0	0,999	0,001
sorghum	23	10	0,1	1	0	0,997	0,003
zinc	21	12	0,583	1	0	0,998	0,002

silver	21	7	0	1	0	0,997	0,003
pet-chem	20	12	0	1	0	0,996	0,004
retail	19	2	0	1	0	0,999	0,001
rapeseed	18	9	0,444	1	0	0,998	0,002
tin	18	12	0,25	1	0	0,997	0,003
wpi	17	9	0,222	1	0	0,997	0,003
strategic-metal	16	11	0	1	0	0,996	0,004
hog	15	6	0,333	1	0	0,999	0,001
orange	15	8	0,5	1	0	0,999	0,001
lead	15	14	0	1	0	0,995	0,005
heat	14	5	0,6	1	0	0,999	0,001
housing	14	3	0,667	1	0	1	0
soy-oil	14	11	0	1	0	0,996	0,004
soy-meal	13	12	0	1	0	0,996	0,004
fuel	13	10	0,2	1	0	0,997	0,003
sunseed	11	5	0	1	0	0,998	0,002
lumber	10	6	0	1	0	0,998	0,002
dmk	10	3	0	1	0	0,999	0,001
lei	9	2	0	1	0	0,999	0,001
tea	9	4	0	1	0	0,999	0,001
nickel	8	1	0	1	0	1	0
oat	7	6	0	1	0	0,998	0,002
l-cattle	6	2	0	1	0	0,999	0,001
sun-oil	5	2	0	1	0	0,999	0,001
rape-oil	5	3	0	1	0	0,999	0,001
groundnut	5	4	0	1	0	0,999	0,001
income	5	5	0	1	0	0,998	0,002
platinum	5	6	0	1	0	0,998	0,002
coconut	4	2	0	1	0	0,999	0,001
jet	4	1	0	1	0	1	0
coconut-oil	4	3	0	1	0	0,999	0,001
propane	3	3	0	1	0	0,999	0,001
instal-debt	3	1	0	1	0	1	0
potato	3	3	0	1	0	0,999	0,001
rand	2	1	0	1	0	1	0
nzdlr	2	2	0	1	0	0,999	0,001
dfl	2	1	0	1	0	1	0
copra-cake	2	1	0	1	0	1	0
palmkernel	2	1	0	1	0	1	0
naphtha	2	4	0	1	0	0,999	0,001
palladium	2	1	0	1	0	1	0
lin-oil	1	1	0	1	0	1	0
nkr	1	1	0	1	0	1	0
rye	1	1	0	1	0	1	0
groundnut-oil	1	1	0	1	0	1	0
castor-oil	1	1	0	1	0	1	0
sun-meal	1	1	0	1	0	1	0
cotton-oil	1	2	0	1	0	0,999	0,001

APPENDIX 4 – RESULTAT TFXIDF-VIKTNING

Nedanstående tabell återger klassificerarens prestanda vid respektive klass. För att vikta termerna användes här tfxidf. Värdena är avrundade med tre decimaler. Antalet dokument för träning och testning är 7,068 och 2,745. Antalet testklasser är 89. Stoppordsborttagning samt suffixstripping med Porter stemmer har genomförts. Värdet på k samt tröskelvärdet är 45 och 0,3. Document frequency Thresholding användes för att reducera dimensioner. Alla termer med dokumentfrekvens under 5 avlägsnades, vilket gav 4,495 unika termer. Micro average breakeven var 0,80.

Topic	*Nof train*	*Nof test*	*Recall*	*Precision*	*Fallout*	*Accuray*	*Error*
earn	2709	1045	0,994	0,835	0,121	0,923	0,077
acq	1488	643	0,916	0,944	0,017	0,968	0,032
money-fx	461	142	0,944	0,62	0,032	0,967	0,033
grain	395	134	0,866	0,72	0,017	0,977	0,023
crude	351	161	0,932	0,777	0,017	0,98	0,02
trade	337	112	0,83	0,694	0,016	0,978	0,022
interest	289	102	0,833	0,708	0,013	0,981	0,019
wheat	198	66	0,833	0,579	0,015	0,981	0,019
ship	192	85	0,8	0,81	0,006	0,988	0,012
corn	161	48	0,729	0,686	0,006	0,989	0,011
sugar	118	35	0,8	0,933	0,001	0,997	0,003
oilseed	117	44	0,409	0,857	0,001	0,989	0,011
coffee	110	27	0,852	0,92	0,001	0,998	0,002
dlr	96	31	0,806	0,455	0,011	0,987	0,013
gold	94	28	0,75	0,6	0,005	0,992	0,008
gnp	92	34	0,824	0,8	0,003	0,995	0,005
money-supply	87	23	0,609	0,778	0,001	0,995	0,005
veg-oil	86	37	0,514	0,95	0	0,993	0,007
livestock	73	24	0,625	0,833	0,001	0,996	0,004
soybean	73	29	0,31	1	0	0,993	0,007
nat-gas	72	29	0,448	0,565	0,004	0,991	0,009
bop	62	28	0,571	0,593	0,004	0,992	0,008
cpi	60	26	0,577	0,469	0,006	0,99	0,01
cocoa	50	15	0,867	0,765	0,001	0,998	0,002
carcass	50	18	0,556	0,769	0,001	0,996	0,004
reserves	48	14	0,214	1	0	0,996	0,004
copper	47	17	0,824	1	0	0,999	0,001
jobs	41	18	0,5	1	0	0,997	0,003
iron-steel	40	12	0,667	0,727	0,001	0,997	0,003
cotton	38	20	0,45	1	0	0,996	0,004
yen	36	12	0	1	0	0,996	0,004
rubber	35	12	0,667	0,889	0	0,998	0,002
ipi	35	10	0,3	1	0	0,997	0,003
rice	35	24	0,125	0,75	0	0,992	0,008
barley	33	12	0,5	1	0	0,998	0,002
alum	33	20	0,4	0,889	0	0,995	0,005
gas	31	14	0,143	1	0	0,996	0,004
meal-feed	30	18	0	1	0	0,993	0,007
palm-oil	29	10	0,6	1	0	0,999	0,001
sorghum	23	10	0	1	0	0,996	0,004
zinc	21	12	0,25	1	0	0,997	0,003

silver	21	7	0	1	0	0,997	0,003
pet-chem	20	12	0	1	0	0,996	0,004
retail	19	2	0	1	0	0,999	0,001
rapeseed	18	9	0,444	1	0	0,998	0,002
tin	18	12	0,333	1	0	0,997	0,003
wpi	17	9	0,333	1	0	0,998	0,002
strategic-metal	16	11	0	1	0	0,996	0,004
hog	15	6	0,333	1	0	0,999	0,001
orange	15	8	0,5	1	0	0,999	0,001
lead	15	14	0	1	0	0,995	0,005
heat	14	5	0,6	1	0	0,999	0,001
housing	14	3	0,333	1	0	0,999	0,001
soy-oil	14	11	0	1	0	0,996	0,004
soy-meal	13	12	0	1	0	0,996	0,004
fuel	13	10	0,2	1	0	0,997	0,003
sunseed	11	5	0	1	0	0,998	0,002
lumber	10	6	0	1	0	0,998	0,002
dmk	10	3	0	1	0	0,999	0,001
lei	9	2	0	1	0	0,999	0,001
tea	9	4	0	1	0	0,999	0,001
nickel	8	1	0	1	0	1	0
oat	7	6	0	1	0	0,998	0,002
l-cattle	6	2	0	1	0	0,999	0,001
sun-oil	5	2	0	1	0	0,999	0,001
rape-oil	5	3	0	1	0	0,999	0,001
groundnut	5	4	0	1	0	0,999	0,001
income	5	5	0	1	0	0,998	0,002
platinum	5	6	0	1	0	0,998	0,002
coconut	4	2	0	1	0	0,999	0,001
jet	4	1	0	1	0	1	0
coconut-oil	4	3	0	1	0	0,999	0,001
propane	3	3	0	1	0	0,999	0,001
instal-debt	3	1	0	1	0	1	0
potato	3	3	0	1	0	0,999	0,001
rand	2	1	0	1	0	1	0
nzdlr	2	2	0	1	0	0,999	0,001
dfl	2	1	0	1	0	1	0
copra-cake	2	1	0	1	0	1	0
palmkernel	2	1	0	1	0	1	0
naphtha	2	4	0	1	0	0,999	0,001
palladium	2	1	0	1	0	1	0
lin-oil	1	1	0	1	0	1	0
nkr	1	1	0	1	0	1	0
rye	1	1	0	1	0	1	0
groundnut-oil	1	1	0	1	0	1	0
castor-oil	1	1	0	1	0	1	0
sun-meal	1	1	0	1	0	1	0
cotton-oil	1	2	0	1	0	0,999	0,001